Hands-on Activity 4.3	
Sorting and Searching Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic & Design	**Date Performed: 9/16/2025**
Section: CPE11S1	**Date Submitted: 9/18/2025**
Name(s): Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**
Output	
Supplementary Activity	

1. Write a program that asks for a number from the user and prints which day of the week that number corresponds to. The days are indexed from 0 (Sunday) to 6 (Saturday). Before the program gets a value from the array, it must first check if the given day is greater than or equal to zero and less than 7. If not, it should print the message: "Error, no such day." Your version of the program must print the same result as the expected output.

Sample Input and Output:

Example input 0

Example output: Sunday

Example input 5

Example output: Friday

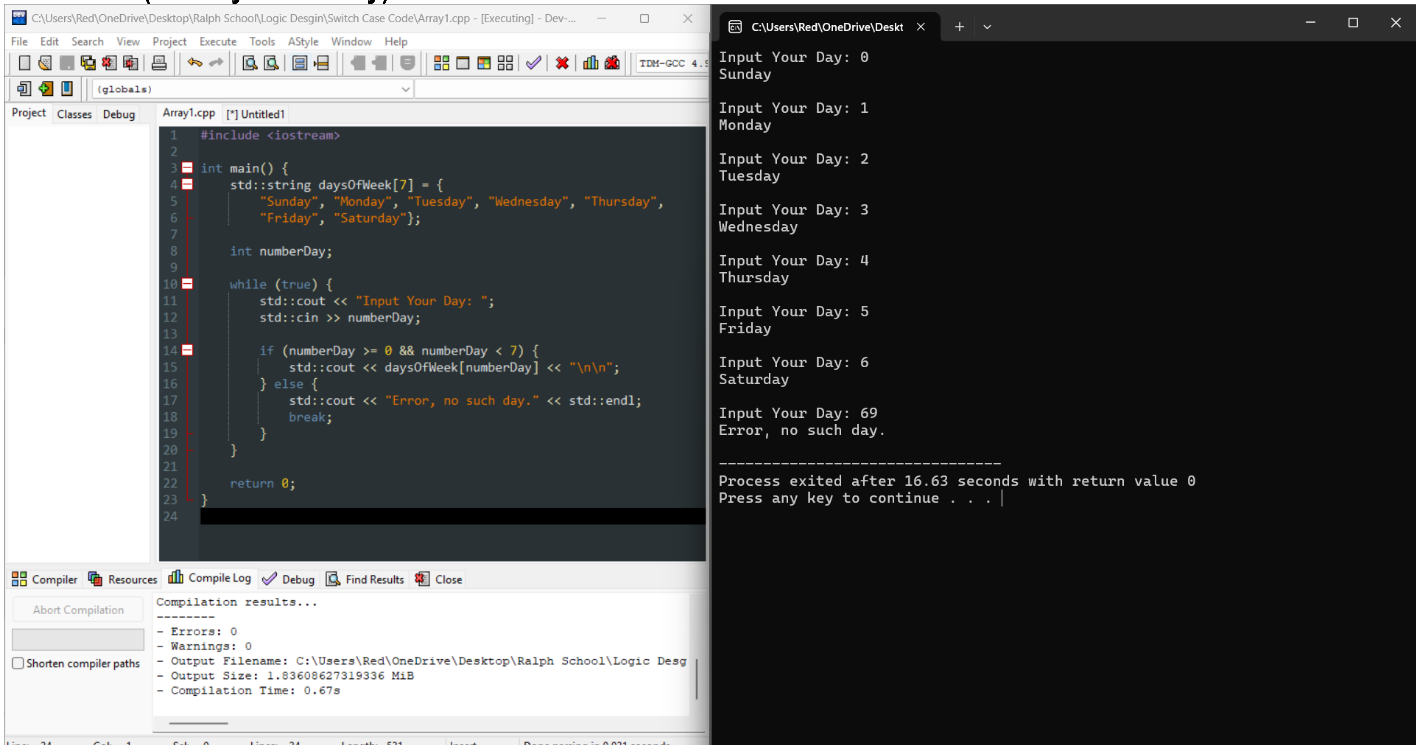
Example input 12

Example output: Error, no such day.

CODE:

```
#include <iostream>

int main() {
    std::string daysOfWeek[7] = {
        "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
                        "Friday", "Saturday"};

    int numberDay;

    while (true) {
        std::cout << "Input Your Day: ";
        std::cin >> numberDay;

        if (numberDay >= 0 && numberDay < 7) {
            std::cout << daysOfWeek[numberDay] << "\n\n";
        } else {
            std::cout << "Error, no such day." << std::endl;
            break;
        }
    }

    return 0;
}
```

RESULTS: (Sunday - Saturday)



C:\Users\Red\OneDrive\Desktop\Ralph School\Logic Desgin\Switch Case Code\Array1.cpp - [Executing] - Dev-...
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9
(globals)
Project Classes Debug
Array1.cpp [*] Untitled1
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
C:\Users\Red\OneDrive\Deskt

```cpp
#include <iostream>

int main() {
    std::string daysOfWeek[7] = {
        "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
        "Friday", "Saturday"};

    int numberDay;

    while (true) {
        std::cout << "Input Your Day: ";
        std::cin >> numberDay;

        if (numberDay >= 0 && numberDay < 7) {
            std::cout << daysOfWeek[numberDay] << "\n\n";
        } else {
            std::cout << "Error, no such day." << std::endl;
            break;
        }
    }

    return 0;
}
```

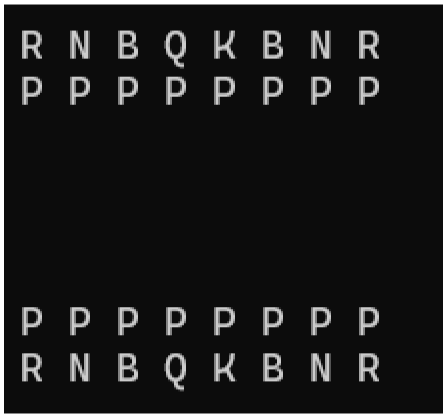
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Red\OneDrive\Desktop\Ralph School\Logic Desg
- Output Size: 1.83608627319336 MiB
- Compilation Time: 0.67s
```

```
Input Your Day: 0
Sunday

Input Your Day: 1
Monday

Input Your Day: 2
Tuesday

Input Your Day: 3
Wednesday

Input Your Day: 4
Thursday

Input Your Day: 5
Friday

Input Your Day: 6
Saturday

Input Your Day: 69
Error, no such day.

--------------------------------
Process exited after 16.63 seconds with return value 0
Press any key to continue . . .
```

How it works:
First we have to make sure that everything in our array is declared as a string, daysofWeek is the name of our array and [7] is the 7 elements of the array indexed from 0 to 6. Next, we have the user input where we declared numberDay; as an int, it prints out “Input your Day: “ then it will ask for the value of numberDay. Now for the while-loop, this is where it will keep asking to input your day until you put an invalid day that isn’t in the array. For our if statement if numberDay >= 0 this makes sure that the day isn’t a negative integer then && to make sure that it also follows the numberDay < 7.So, when the user enters a number, it matches directly to the index. Example: numberDay = 0 and that corresponds with Sunday. Our else statement is where it breaks if the numberDay is not between 0 and 6, so it will print out an error message and stops the program (this is why numberDay < 7**)** .

2. Write a program that creates a chessboard, sets all the pieces on it and then displays the contents of the board. Create a two-dimensional array, fill it with data and print a letter when a piece is on the field and a space when there is no piece. Store one letter for one piece. For now, we don't need any information about the color of the pieces. The starting positions (with letters which symbolize each piece) for all pieces are: The rooks (R) are placed on the outside corners, right and left edge (white on the 1st and black on the 8th line). The knights (N) are placed immediately inside of the rooks. The bishops (B) are placed immediately inside of the knights. The queen (Q) is placed on the central square of the same color as that of the player: white queen on the white square and black queen on the black square. Both stand on the d rank: white queen on the d1 field and black queen on the d8 field. The king (K) takes the vacant spot next to the queen. The pawns (P - not the official symbol, but you need to print something) are placed one square in front of all of the

other pieces. Your version of the program must print the same result as the expected output.

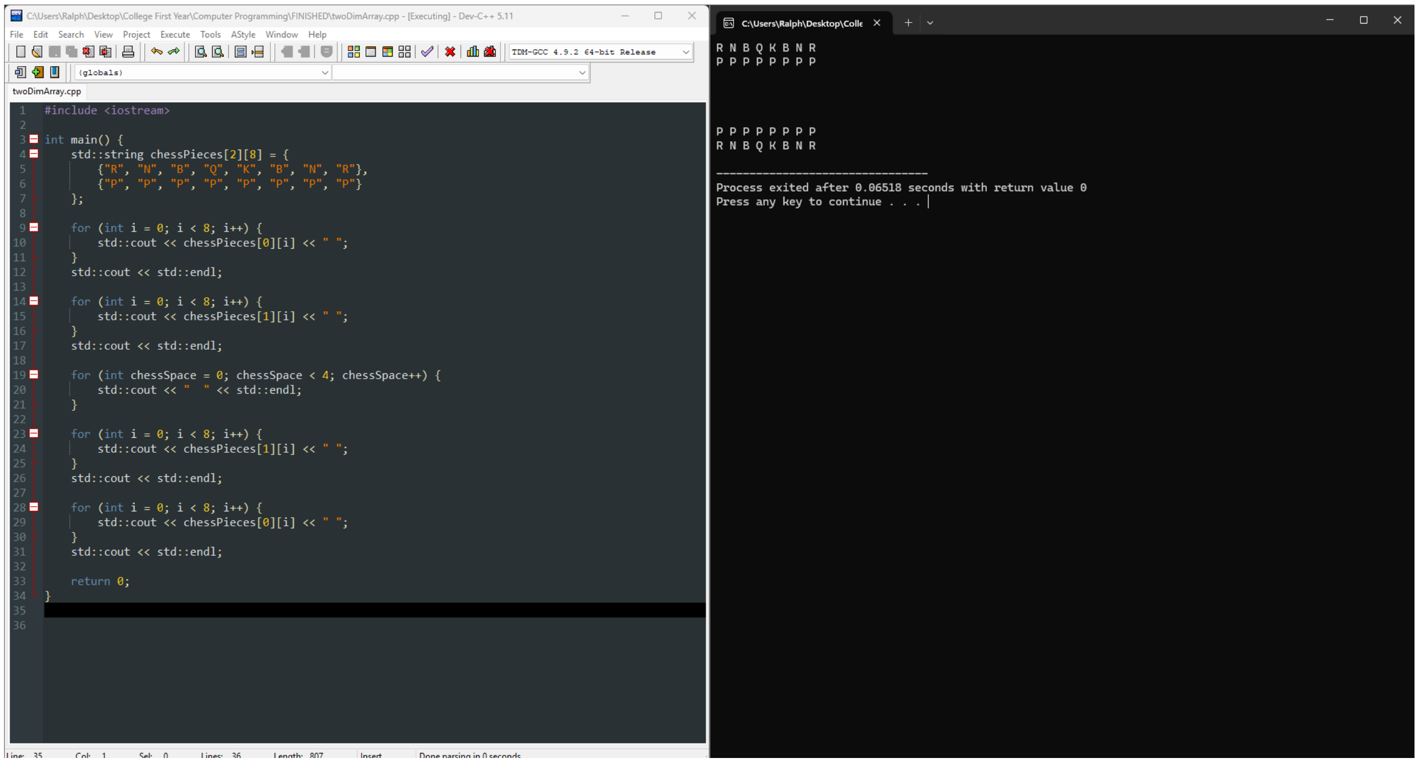


```
R N B Q K B N R
P P P P P P P P




P P P P P P P P
R N B Q K B N R
```

CODE:

```
#include <iostream>

int main() {
    std::string chessPieces[2][8] = {
        {"R", "N", "B", "Q", "K", "B", "N", "R"},
        {"P", "P", "P", "P", "P", "P", "P", "P"}
    };

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[0][i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[1][i] << " ";
    }
    std::cout << std::endl;

    for (int chessSpace = 0; chessSpace < 4; chessSpace++) {
        std::cout << "  " << std::endl;
    }

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[1][i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[0][i] << " ";
    }
    std::cout << std::endl;

    return 0;

}
```

RESULT:


C:\Users\Ralph\Desktop\College First Year\Computer Programming\FINISHED\twoDimArray.cpp - [Executing] - Dev-C++ 5.11

twoDimArray.cpp

```cpp
#include <iostream>

int main() {
    std::string chessPieces[2][8] = {
        {"R", "N", "B", "Q", "K", "B", "N", "R"},
        {"P", "P", "P", "P", "P", "P", "P", "P"}
    };

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[0][i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[1][i] << " ";
    }
    std::cout << std::endl;

    for (int chessSpace = 0; chessSpace < 4; chessSpace++) {
        std::cout << "  " << std::endl;
    }

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[1][i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 8; i++) {
        std::cout << chessPieces[0][i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

```
R N B Q K B N R
P P P P P P P P




P P P P P P P P
R N B Q K B N R

--------------------------------
Process exited after 0.06518 seconds with return value 0
Press any key to continue . . .
```

File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)

How it works:

First we have to declare that everything in the array is a string because we will be printing out the pieces of the chessboard. chessPeices is the name of our two-dimensional array [2] meaning 2 rows and [8] meaning 8 elements in those 2 rows.

{"R", "N", "B", "Q", "K", "B", "N", "R"},
{"P", "P", "P", "P", "P", "P", "P", "P"}

Now we are heading to the printing part of the code, the first for-loop that we have here will print out the first row of the chessboard this is where all the major pieces are chessPieces[0][i] and " " will make sure they is a space between the pieces so it wont stick together. It will keep printing until i < 8 because a chessboard is 8 by 8 and this will be the same for the second row of chessboard chessPieces[1][i] where it will print all the pawns. Here we are making the blank space of the chessboard by printing 4 empty rows (again because it is 8x8) it prints out " " then goes to the next line till chessSpace < 4. Now we will be printing the other side of the chessboard by doing the same for-loop as before except this time the pawns goes first then the main pieces come second and that is the end of the code.

Conclusion

I've learned to use string instead of integer when making an array and discovered I could use break; for other things as well and not just switch cases (for the longest time I thought I could use break; for switch cases only). This supplementary activity looks easy, but figuring it out was very confusing. I've gotten a lot of errors trying to make the while-loop work for the first activity, and I'll be honest, the reason why it took me a while was because I'd set the array as an integer first instead of a string for the activity to display the day of the week. The second supplementary activity was quite easy after figuring out how to print the two-dimensional loop, and it took me some time to figure out how to display a specific row, but after some trial and error, I managed to get the hang of it. The activity improved my understanding of for-loops and while-loops because I've never known what the difference between them was, and that's what made me confused when I was doing the other activities. In this activity, I'd think I did pretty well because last time it took me hours (around 3) just to code it and another 2 hours trying to explain it. I've become faster, and when I look at the code, my brain doesn't instantly shut off anymore, but again, there are still so many things I need to improve because without help from the internet, I wouldn't be able to complete this hands-on activity. I still have to improve logically, my code reading skills, and my efficiency, but other than that, I could say I've improved.