2D Arrays

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

Working Example

Suppose, that you want to save the number of cars in stock. The company sells five types of cars in five different colours.

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1

Working Example

How can we Save this information in memory?

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1

Parallel Arrays

We can make parallel arrays to store this information.

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1

Parallel Arrays

We can make parallel arrays to store this information.

```
                          Red  Black  Brown  Blue  Gray
                           0     1      2     3     4
int Suzuki[5] = {10, 7, 12, 10, 4};
int Toyota[5] = {18, 11, 15, 17, 2};
int Nissan[5] = {23, 19, 12, 16, 14};
int BMW[5]    = {7, 12, 16, 0, 2};
int Audi[5]   = {3, 5, 6, 2, 1};
```

Parallel Arrays

We can make parallel arrays to store this information.
What will be the output of cout << Toyota[2];

```
                        Red  Black  Brown  Blue  Gray
                         0     1      2     3     4
int Suzuki[5] = {10, 7, 12, 10, 4};
int Toyota[5] = {18, 11, 15, 17, 2};
int Nissan[5] = {23, 19, 12, 16, 14};
int BMW[5]    = {7, 12, 16, 0, 2};
int Audi[5]   = {3, 5, 6, 2, 1};
```

Parallel Arrays

We can make parallel arrays to store this information.
What will be the output of cout << Audi[4];

```
                        Red  Black  Brown  Blue  Gray
                         0     1      2     3     4
int Suzuki[5] = {10, 7, 12, 10, 4};
int Toyota[5] = {18, 11, 15, 17, 2};
int Nissan[5] = {23, 19, 12, 16, 14};
int BMW[5]    = {7, 12, 16, 0, 2};
int Audi[5]   = {3, 5, 6, 2, 1};
```

Parallel Arrays

Do you see any problem in this approach?

```
                          Red  Black  Brown  Blue  Gray
                           0     1      2     3     4
int Suzuki[5] = {10, 7, 12, 10, 4};
int Toyota[5] = {18, 11, 15, 17, 2};
int Nissan[5] = {23, 19, 12, 16, 14};
int BMW[5]    = {7, 12, 16, 0, 2};
int Audi[5]   = {3, 5, 6, 2, 1};
```

Parallel Arrays

It's not an effective solution.
We had to declare 5 separate arrays with different names to store the information which is related to the same kind of objects and have same datatype.

```
                          Red  Black  Brown  Blue  Gray
                           0     1      2     3     4
int Suzuki[5] = {10, 7, 12, 10, 4};
int Toyota[5] = {18, 11, 15, 17, 2};
int Nissan[5] = {23, 19, 12, 16, 14};
int BMW[5]    = {7, 12, 16, 0, 2};
int Audi[5]   = {3, 5, 6, 2, 1};
```

Better Data Structure

We have a better data structure that can organize this information in more efficient and connected manner.

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1

Matrix

This Data Structure came from Math and It is called **Matrix**.

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1

Matrix

This information can be represented in the form of **rows** and **columns**.

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

2D Array

C++ provides 2D arrays that can store similar types of information present in more than one row.

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

2D Array

There are 5 rows and 5 columns.

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Initialization

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
cout << cars[0][0];
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
cout << cars[0][1];
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
cout << cars[0][2];
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
cout << cars[2][0];
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
cout << cars[4][2];
```

		Red 0	Black 1	Brown 2	Blue 3	Gray 4
Suzuki	0	10	7	12	10	4
Toyota	1	18	11	15	17	2
Nissan	2	23	19	12	16	14
BMW	3	7	12	16	0	2
Audi	4	3	5	6	2	1

Access Data

We have accessed data cell individually. Can you write a program that print all the values of the 2D array using loop ?

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

Access Data

To print all data of 2D array, we can use following program.

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
main()
{
    for(int x = 0; x < 5; x = x + 1)
    {
        for(int y = 0; y < 5; y = y + 1)
        {
            cout << cars[x][y] << "\t";
        }
        cout << endl;
    }
}
```

Access Data

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

The program will show following output on the screen.

```
main()
{
    for(int x = 0; x < 5; x = x + 1)
    {
        for(int y = 0; y < 5; y = y + 1)
        {
            cout << cars[x][y] << "\t";
        }
        cout << endl;
    }
}
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
10      7       12      10      4
18      11      15      17      2
23      19      12      16      14
7       12      16      0       2
3       5       6       2       1

C:\C++>
```

Activity 01

Write a Function that returns the sum of all the colors of all the cars.

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

Activity 01

Write a Function that returns the sum of all the colors of all the cars.

```
int sumCars()
{


}
```

Activity 01: Solution

Write a Function that returns the sum of all the colors of all the cars.

```
int sumCars()
{
    int sum = 0;
    for(int row = 0; row < 5; row++)
    {
        for(int col =0; col < 5; col++)
        {
            sum = sum + cars[row][col];
        }
    }
    return sum;
}
```

Activity 02

Write a Function that returns the largest quantity of cars from all the cars.

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

Activity 02

Write a Function that returns the largest quantity of cars from all the cars.

```
int largestQuantity()
{


}
```

Activity 02: Solution

Write a Function that returns the largest quantity of cars from all the cars.

```
int largestQuantity()
{
    int maximum = cars[0][0];
    for (int row = 0; row < 5; row++)
    {
        for (int col = 0; col < 5; col++)
        {
            if (maximum < cars[row][col])
            {
                maximum = cars[row][col];
            }
        }
    }
    return maximum;
}
```

Activity 03

A batch contains cars which has same company and same color and the number of cars in batch is always greater than 15. Write a Function that returns the total number of batches within the data.

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

Activity 03

A batch contains cars which has same company and same color and the number of cars in batch is always greater than 15. Write a Function that returns the total number of batches within the data.

```
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                 };
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
Total count: 6
C:\C++>
```

Activity 03

A batch contains cars which has same company and same color and the number of cars in batch is always greater than 15. Write a Function that returns the total number of batches within the data.

```
#include <iostream>
using namespace std;
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}};


main()
{
    int total;
    total = batchCount();
    cout << total;
}
```

Activity 03: Solution

A batch contains cars which has same company and same color and the number of cars in batch is always greater than 15. Write a program that list down the total number of batches within the data

```
int batchCount()
{
    int count = 0;
    for (int x = 0; x < 5; x = x + 1)
    {
        for (int y = 0; y < 5; y = y + 1)
        {
            if (cars[x][y] > 15)
            {
                count = count + 1;
            }
        }
    }
    return count;
}
```

Activity 03: Solution

A batch contains cars which has same company and same color and the number of cars in batch is always greater than 15. Write a program that list down the total number of batches within the data

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
Total count: 6
C:\C++>
```

```
int batchCount()
{
    int count = 0;
    for (int x = 0; x < 5; x = x + 1)
    {
        for (int y = 0; y < 5; y = y + 1)
        {
            if (cars[x][y] > 15)
            {
                count = count + 1;
            }
        }
    }
    return count;
}
```

Learning Objective

Understand the difference between 1D and 2D arrays.

Learning Objective

Write C++ Program that creates a 2D array and store similar data consisting of multiple rows using a single name.

Conclusion

- Limitation of the 1D array is that it contains only 1 row for storing similar records whereas a 2D array can store similar records consisting of multiple rows using a single name. The general form of declaring a 2D array is:

  dataType arrayName[rowNumber][columnNumber];

- The general form (syntax) used for accessing a 2D array component is:

  arrayName[rowNumber][columnNumber];

where rowNumber specifies the row index and columnNumber specifies the column index.

- Just like in 1D array, the row and column index in 2D array also starts from 0.

Self Assessment

1. Tic-Tac-toe is a paper and pencil game for two players. There are two symbols that are used in this game: X and O, both of the players can choose either of them. They will take their turn to mark spaces in a 3x3 grid. The player will win if he/she succeeds in marking three of his/her symbol in a diagonal, horizontal or vertical row

Assessment

1. The const keyword specifies that a variable's value is constant and tells the compiler to prevent the programmer from modifying it. Write a C++ function to Identify the Winner from this board configuration.

```
#include <iostream>
using namespace std;
const int gridSize = 3;
char board[gridSize][gridSize] = {{'O', 'O', 'X'},
                                  {'O', 'O', 'X'},
                                  {'X', 'X', 'O'}
                                  };
bool isWinner(char symbol)
{
    // Write your Code here
}
main()
{
    if (isWinner('X'))
    {
        cout << "Winner is Player X";
    }
    else if (isWinner('O'))
    {
        cout << "Winner is Player 0";
    }
}
```