



2D Arrays in C#



Working Example: Vision

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


|| Solution

In C++, we initialized the 2D array as follows.

```
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}  
};
```

Solution

In C#, we initialize the 2D array as follows.



```
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}  
};
```

```
int [,] cars = {{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}  
};
```

Solution

In *C#*, if we just want to declare the 2D array then it is as follows.

```
int [,] cars = new int[5,5];
```

Accessing the Elements

In C++, we accessed the elements of the 2D array as follows.

	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

Accessing the Elements

In C++, we accessed the elements of the 2D array as follows.

```
cars[0][2];
```

	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

Accessing the Elements

In C#, we access the elements of the 2D array as follows.

✗ `cars[0][2];`

`cars[0,2];` ✓

	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: Vision

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

	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

Solution

```
static int printSum(int [,] cars)
{
    int sum = 0;
    for (int x = 0; x < 5; x++)
    {
        for (int y = 0; y < 5; y++)
        {
            sum = sum + cars[x, y];
        }
    }
    return sum;
}
```

```
static void Main(string[] args)
{
    int [,] cars = {
        { 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}
    };

    int sum;
    sum = printSum(cars);
    Console.WriteLine("Sum is: {0}", sum);
    Console.Read();
}
```



Game Development in C#



PacMan Game

```
%%%%%%%%%
%       %
%       %
%       %
%   P   %
%       %
%       %
%       %
%       %
%%%%%%%%%
```


PacMan Game: Maze

```
char[,] maze = new char[10,10] {  
    { '%', '%', '%', '%', '%', '%', '%', '%', '%', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%'},  
    { '%', '%', '%', '%', '%', '%', '%', '%', '%', '%'},  
};
```

Solution C++

In C++, we used `gotoxy(x,y)`

To display something on the specific location on the console.



```
void movePacmanRight()
{
    if (maze[pacmanX][pacmanY + 1] == ' ' || maze[pacmanX][pacmanY + 1] == '.')
    {
        maze[pacmanX][pacmanY] = ' ';
        gotoxy(pacmanY, pacmanX);
        cout << " ";
        pacmanY = pacmanY + 1;
        gotoxy(pacmanY, pacmanX);
        cout << "P";
    }
}
```

Solution C#: MovePacManUp

In C#, we will use `Console.SetCursorPosition(x,y)`
To display something on the specific location on the console.

```
static void movePacManUp(char[,] maze, ref int pacmanX, ref int pacmanY)
{
    if (maze[pacmanX - 1,pacmanY] == ' ' || maze[pacmanX - 1,pacmanY] == '.')
    {
        maze[pacmanX,pacmanY] = ' ';
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write(" ");
        pacmanX = pacmanX - 1;
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write("P");
    }
}
```

Solution C#: MovePacManDown

In C#, we will use `Console.SetCursorPosition(x,y)`
To display something on the specific location on the console.

```
static void movePacManDown(char[,] maze, ref int pacmanX, ref int pacmanY)
{
    if (maze[pacmanX + 1,pacmanY] == ' ' || maze[pacmanX + 1,pacmanY] == '.')
    {
        maze[pacmanX,pacmanY] = ' ';
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write(" ");
        pacmanX = pacmanX + 1;
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write("P");
    }
}
```


Solution C#: MovePacManLeft

In C#, we will use `Console.SetCursorPosition(x,y)`
To display something on the specific location on the console.

```
static void movePacManLeft(char[,] maze, ref int pacmanX, ref int pacmanY)
{
    if (maze[pacmanX,pacmanY - 1] == ' ' || maze[pacmanX,pacmanY - 1] == '.')
    {
        maze[pacmanX,pacmanY] = ' ';
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write(" ");
        pacmanY = pacmanY - 1;
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write("P");
    }
}
```

Solution C#: MovePacManRight


In C#, we will use `Console.SetCursorPosition(x,y)`
To display something on the specific location on the console.

```
static void movePacManRight(char[,] maze, ref int pacmanX, ref int pacmanY)
{
    if (maze[pacmanX,pacmanY + 1] == ' ' || maze[pacmanX,pacmanY + 1] == '.')
    {
        maze[pacmanX,pacmanY] = ' ';
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write(" ");
        pacmanY = pacmanY + 1;
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write("P");
    }
}
```

Solution C#: MovePacManRight

In C#, we will use `Console.SetCursorPosition(x,y)`
To display something on the specific location on the console.

```
static void movePacManRight(char[,] maze, ref int pacmanX, ref int pacmanY)
{
    if (maze[pacmanX,pacmanY + 1] == ' ' || maze[pacmanX,pacmanY + 1] == '.')
    {
        maze[pacmanX,pacmanY] = ' ';
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write(" ");
        pacmanY = pacmanY + 1;
        Console.SetCursorPosition(pacmanY, pacmanX);
        Console.Write("P");
    }
}
```



Important thing to note here is that we are passing the X and Y coordinates of Pacman by **reference** to this function, so the changes are done in the single variable.

Solution C++

```
main() {  
    bool gameRunning = true;  
    while (gameRunning) {  
        Sleep(100);  
        system("CLS");  
        printMaze();  
        printScore();  
        if (GetAsyncKeyState(VK_LEFT)) {  
            movePacmanLeft();  
        }  
        if (GetAsyncKeyState(VK_RIGHT)) {  
            movePacmanRight();  
        }  
        if (GetAsyncKeyState(VK_UP)) {  
            movePacmanUP();  
        }  
        if (GetAsyncKeyState(VK_DOWN)) {  
            movePacmanDown();  
        }  
        if (GetAsyncKeyState(VK_ESCAPE)) {  
            gameRunning = false; } }  
}
```

Solution C++

In C++, we used
`GetAsyncKeyState(KeyCode)`
To detect if the key was
pressed or not.




```
main() {  
    bool gameRunning = true;  
    while (gameRunning) {  
        Sleep(100);  
        system("CLS");  
        printMaze();  
        printScore();  
        if (GetAsyncKeyState(VK_LEFT)) {  
            movePacmanLeft();  
        }  
        if (GetAsyncKeyState(VK_RIGHT)) {  
            movePacmanRight();  
        }  
        if (GetAsyncKeyState(VK_UP)) {  
            movePacmanUP();  
        }  
        if (GetAsyncKeyState(VK_DOWN)) {  
            movePacmanDown();  
        }  
        if (GetAsyncKeyState(VK_ESCAPE)) {  
            gameRunning = false; } }  
}
```

Solution C#

In C#, we will use
Keyboard.IsKeyPressed(Key.UpArrow)
To detect if the key is
pressed or not.

```
static void Main(string[] args)
{
    int pacmanX = 4; int pacmanY = 4;
    char[,] maze = new char[10,10] {
        { '%', '%', '%', '%', '%', '%', '%', '%', '%', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '%' },
        { '%', '%', '%', '%', '%', '%', '%', '%', '%', '%' };
    printMaze(maze);
    Console.SetCursorPosition(pacmanY, pacmanX);
    Console.Write("P");
    while (true){
        Thread.Sleep(150);
        if (Keyboard.IsKeyPressed(Key.UpArrow)){
            movePacManUp(maze, ref pacmanX, ref pacmanY);
        }
        if (Keyboard.IsKeyPressed(Key.DownArrow)){
            movePacManDown(maze, ref pacmanX, ref pacmanY);
        }
        if (Keyboard.IsKeyPressed(Key.LeftArrow)){
            movePacManLeft(maze, ref pacmanX, ref pacmanY);
        }
        if (Keyboard.IsKeyPressed(Key.RightArrow)){
            movePacManRight(maze, ref pacmanX, ref pacmanY);
        }
    }
}
```



EZInput Package

In order to use the function

`Keyboard.IsKeyPressed(Key.UpArrow)`

We have to install and include the **EZInput** package first.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using EZInput;
```

EZInput Package

In order to use the function

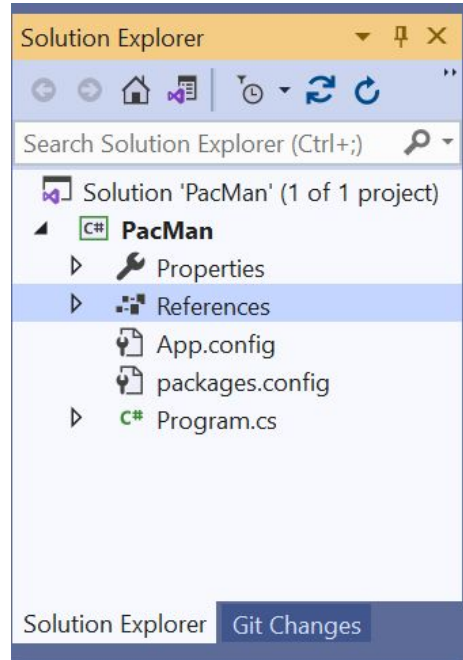
`Keyboard.IsKeyPressed(Key.UpArrow)`

We have to install and include the **EZInput** package first.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using EZInput; ←
```

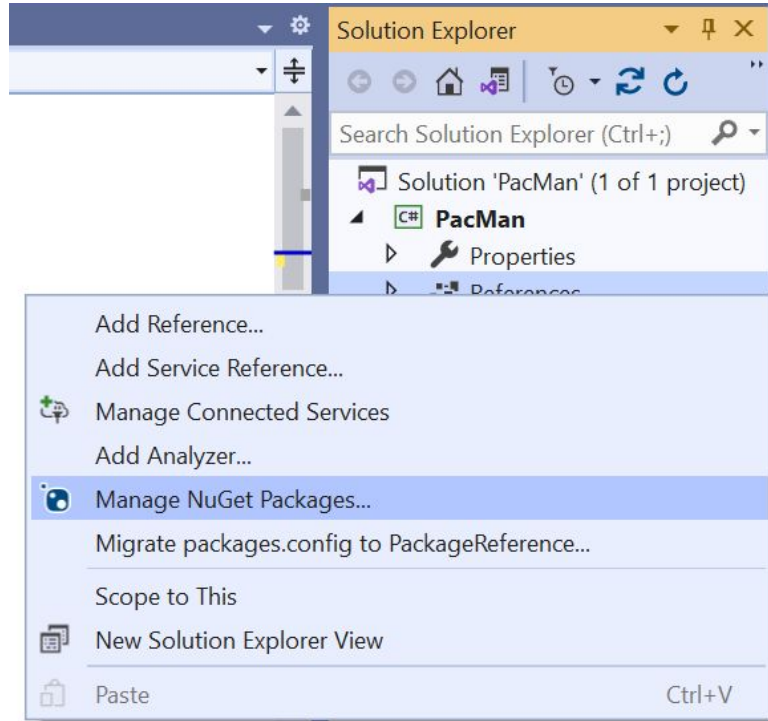

EZInput Package

In order to install the Package, **right click** on the references in Solution Explorer Window



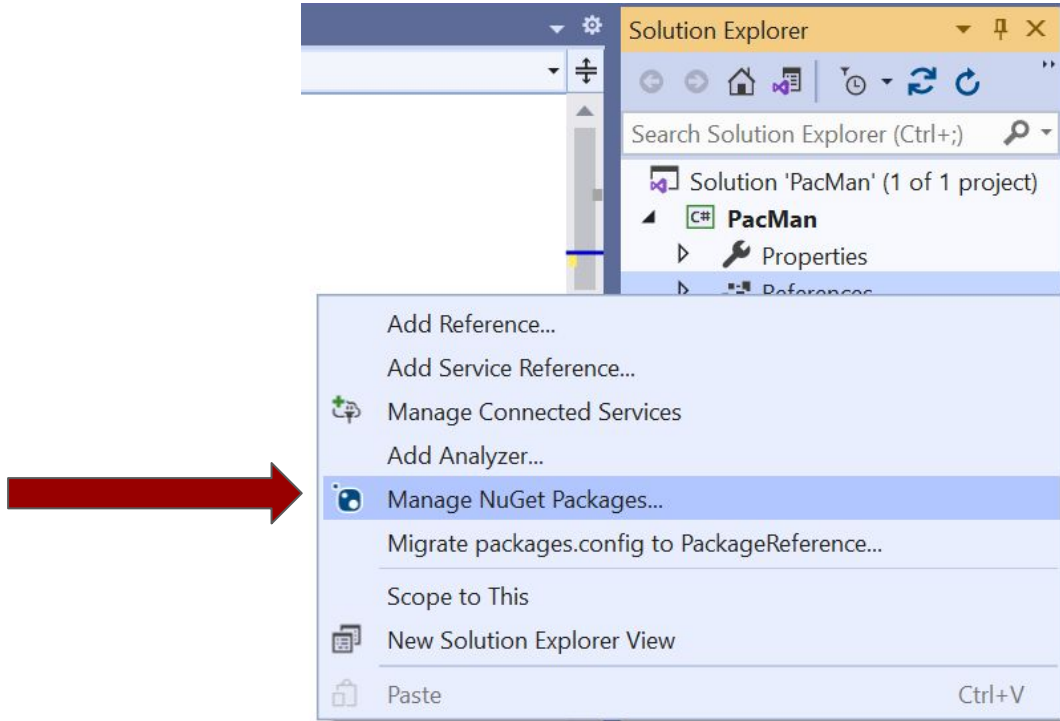
EZInput Package

Then Click on the **Manage NuGet Packages**.



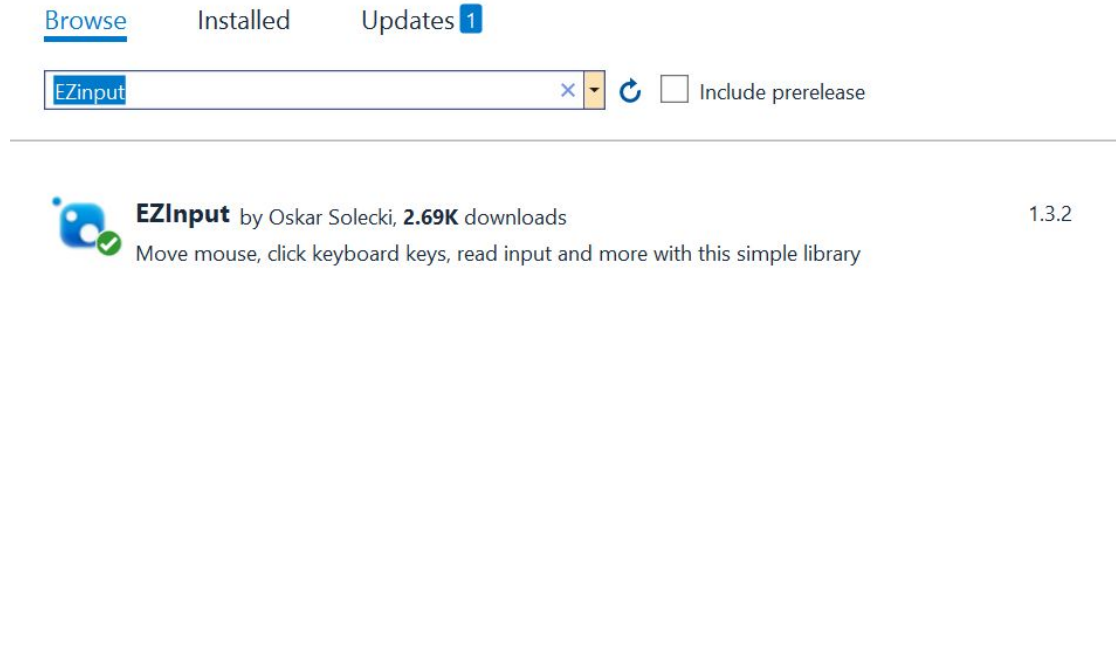
EZInput Package

Then Click on the **Manage NuGet Packages**.



EZInput Package

Search EZInput in the browser and install the package.



EZInput Package

And then include the **EZInput** package using the following instruction.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using EZInput; ←
```

Learning Objective

Write **C#** programs to solve complex 2D problems and convert previous 2D game into C#.



Self Assessment

1. In probability theory, a probability matrix is a matrix such that:

The matrix is a **square matrix** (same number of rows as columns). All entries are probabilities, i.e. numbers between 0 and 1. All rows add up to 1.

The following is an example of a **probability matrix**:

```
[  
  [0.5, 0.5, 0.0],  
  [0.2, 0.5, 0.3],  
  [0.1, 0.2, 0.7]  
]
```



Self Assessment

Note that though all rows add up to 1, there is no restriction on the columns, which may or may not add up to 1.

Write a function that determines if a matrix is a probability matrix or not.

Fun fact:

for most probability matrices M (for example, if M has no zero entries), the matrix powers M^n converge (as n increases) to a matrix where all rows are identical.



Self Assessment

Test Cases:

Input	Output	Explanation
<pre>[[0.5, 0.5, 0.0], [0.2, 0.5, 0.3], [0.1, 0.2, 0.7]] isProbMatrix()</pre>	true	
<pre>[[0.5, 0.5, 0.0], [0.2, 0.5, 0.3]] isProbMatrix()</pre>	false	// Not a square matrix.



Self Assessment

Test Cases:

Input	Output	Explanation
<pre>[[0.5, 0.4], [0.5, 0.6]] isProbMatrix()</pre>	false	// Rows do not add to 1.
<pre>[[2, -1], [-1, 2]] isProbMatrix()</pre>	false	// Entries not between 0 and 1.

