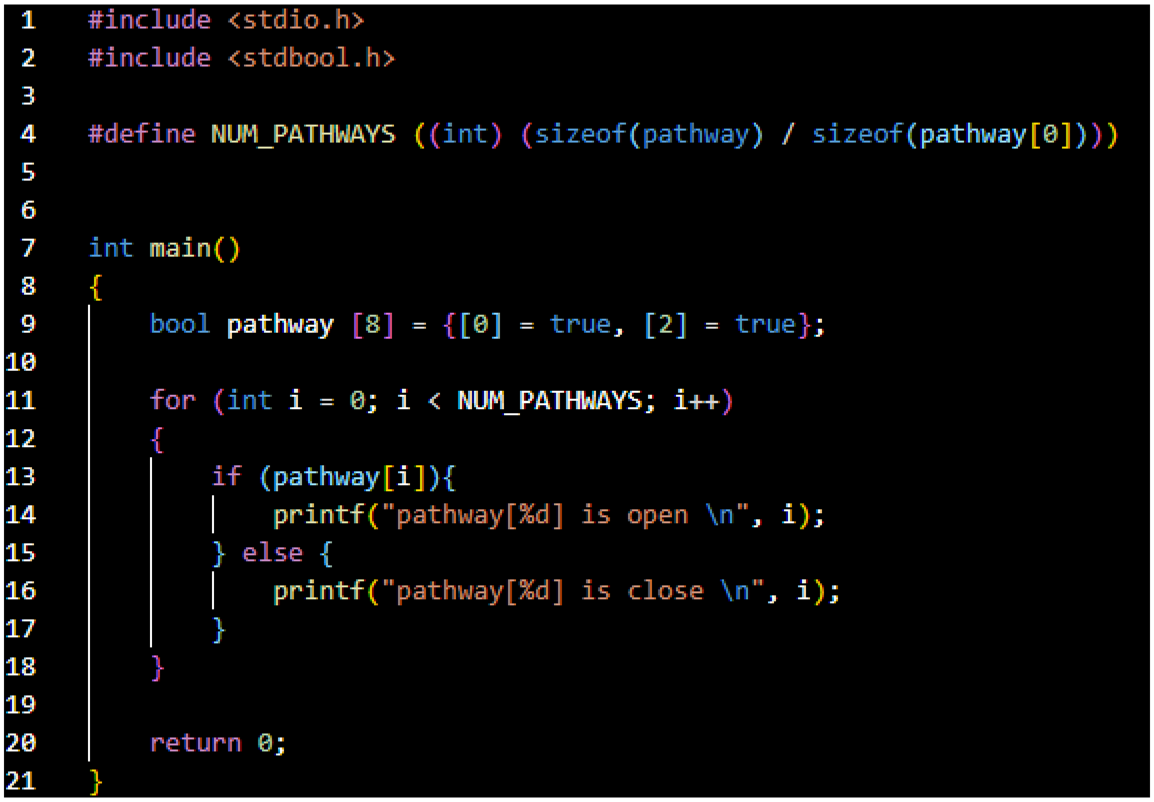
Trifalgar, Michael Rainiken
CMSC 21

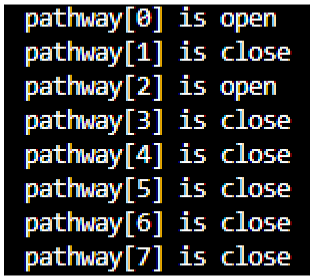
May 8, 2022
Lecture 6-7 Assignment

1a.)
Revised Array with Designated Initializers:



```c
#include <stdio.h>
#include <stdbool.h>

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))


int main()
{
    bool pathway [8] = {[0] = true, [2] = true};

    for (int i = 0; i < NUM_PATHWAYS; i++)
    {
        if (pathway[i]){
            printf("pathway[%d] is open \n", i);
        } else {
            printf("pathway[%d] is close \n", i);
        }
    }

    return 0;
}
```

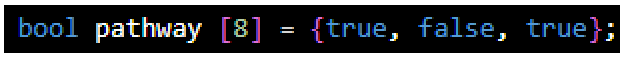
Output:



```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

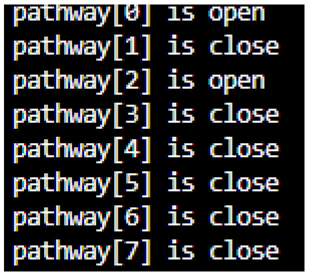
Trifalgar, Michael Rainiken  May 8, 2022
CMSC 21  Lecture 6-7 Assignment

1b.)
Revised Array with no designated initializer:



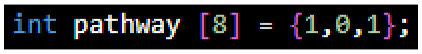
```
bool pathway [8] = {true, false, true};
```

Output:



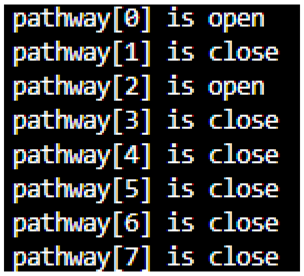
```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

Revised Array if we can replace the bool array with one of type int:



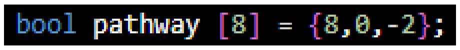
```
int pathway [8] = {1,0,1};
```

Output:

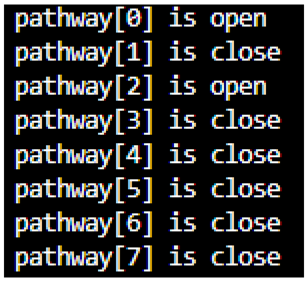


```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

I noticed that the code still produced the same output if the array was type bool and the elements inside it were 1's and 0's. I also checked the output if we replaced the 1's with any non-zero value and it still worked. Any non-zero value is evaluated to true and 0 to false. It makes me wonder if there's an advantage to using boolean instead of just using integers. Are booleans only used to make the code more readable and understandable?



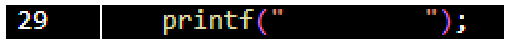
```
bool pathway [8] = {8,0,-2};
```



```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

Trifalgar, Michael Rainiken　　　　　　　　　　May 8, 2022
CMSC 21　　　　　　　　　　　　　　　　　　Lecture 6-7 Assignment

2.)
The first challenge I had while writing this code was printing out the adjacency matrix for the user. Printing out the contents of the 8 x 8 array was fine but I had a lot of trouble trying to align the column labels on the matrix. So I just printed out a couple of white spaces to align the column labels to their values. Here:



```
    printf("        ");
```

I wonder if there's a more elegant way of aligning the letters with their values.

My next problem was trying to print out brackets on charging stations C and D. My initial solution was to create a 2d-array (since character arrays can only have 1 character) so that charging stations C and D can have brackets around them. However, as we travel from station to station, the program prints out points C and D with brackets which is not what's on the sample cases. So I just used an if-else statement to attach a bracket to charging stations C and D even though it takes more lines.

I think creating the array 'is_charging_station[8]' was a cool solution to accommodate for situations that only needed charging stations C and D. My old solution was to use logical operators in the conditional expressions

It looked something like this: if ((i == 3) || (i == 4))

Which I think is fine for smaller matrices like the one in our problem but I think the new solution is cleaner and understandable for larger matrices.

I also just used int instead of bool because I'm more comfortable with using integers for truth values.

There's one thing I noticed from the test case on the pdf file containing the instructions. I noticed that if the user chose point F, that user will travel to point A and B before arriving at C. However, looking at the adjacency matrix, we can clearly see that point F is already adjacent to charging station C so there's no need to travel to other nodes. I also created a diagram on the adjacency matrix. I saw that only points A and points H weren't next to a charging station. So to be more efficient, I included a conditional statement that stops the loop if there's a valid charging station adjacent to the point.

You'll also see that I have iterated backwards when inspecting a node for adjacent charging stations. I specifically made this decision for point A. If the user chose point A, I want the route to be A to B to C to follow the sample cases. If we iterated through the row normally, the next node that the point would have traveled to would be F instead of B. The program however is still completely valid if we iterated through a specific row normally.