Review

- What is the syntax for a macro?
  - #define IDENTIFIER replacement

Review

- Why do we use include guards?
    - to prevent the contents of a file, from being included more than once

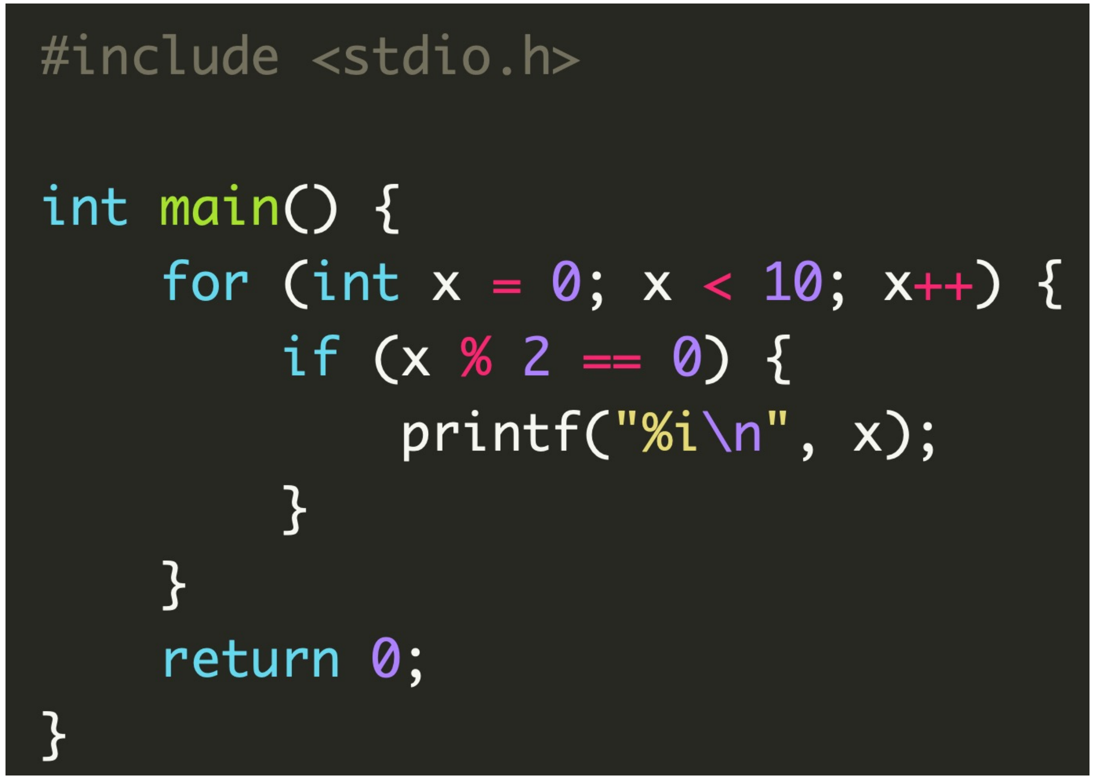
Code Style

Indentation

- In programming languages like C indentation helps visualize nested blocks of code.
- Each indent should be at least 4 spaces
- If you do not want to repeatedly press the spacebar, you can also use the tab key for indentation.
    - This will give you consistent indentation in your program.

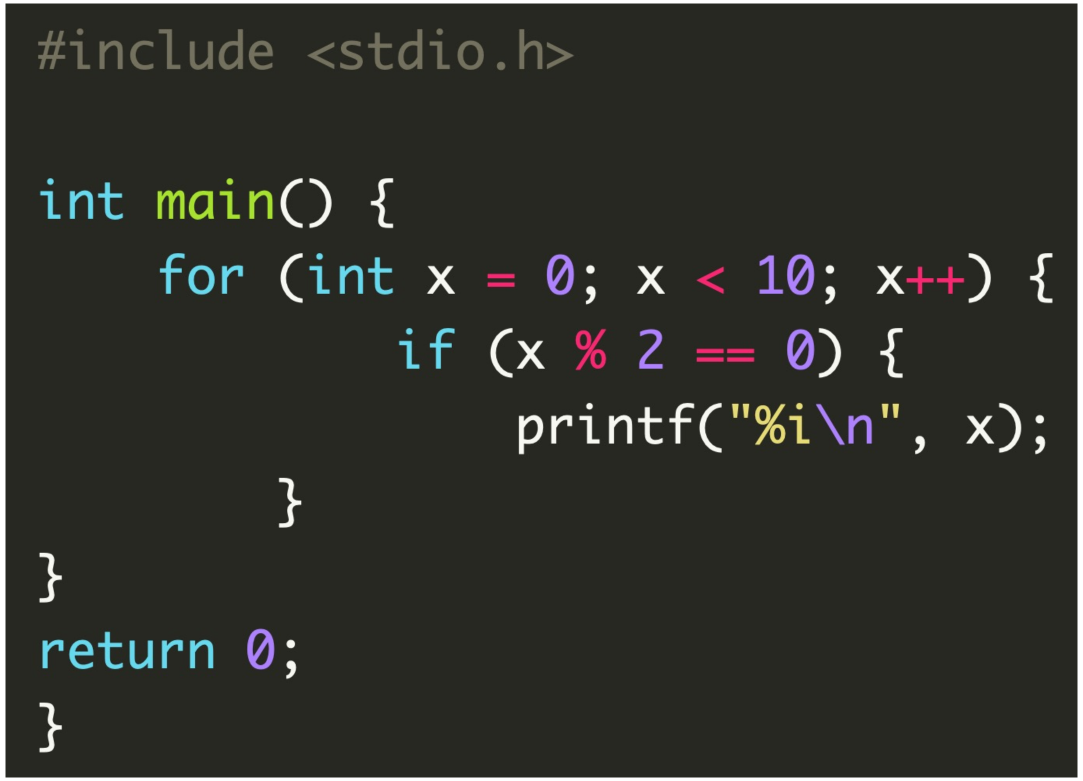
Indentation

DO: Indent with four spaces



```c
#include <stdio.h>

int main() {
    for (int x = 0; x < 10; x++) {
        if (x % 2 == 0) {
            printf("%i\n", x);
        }
    }
    return 0;
}
```

DON'T: Mix different indentation sizes



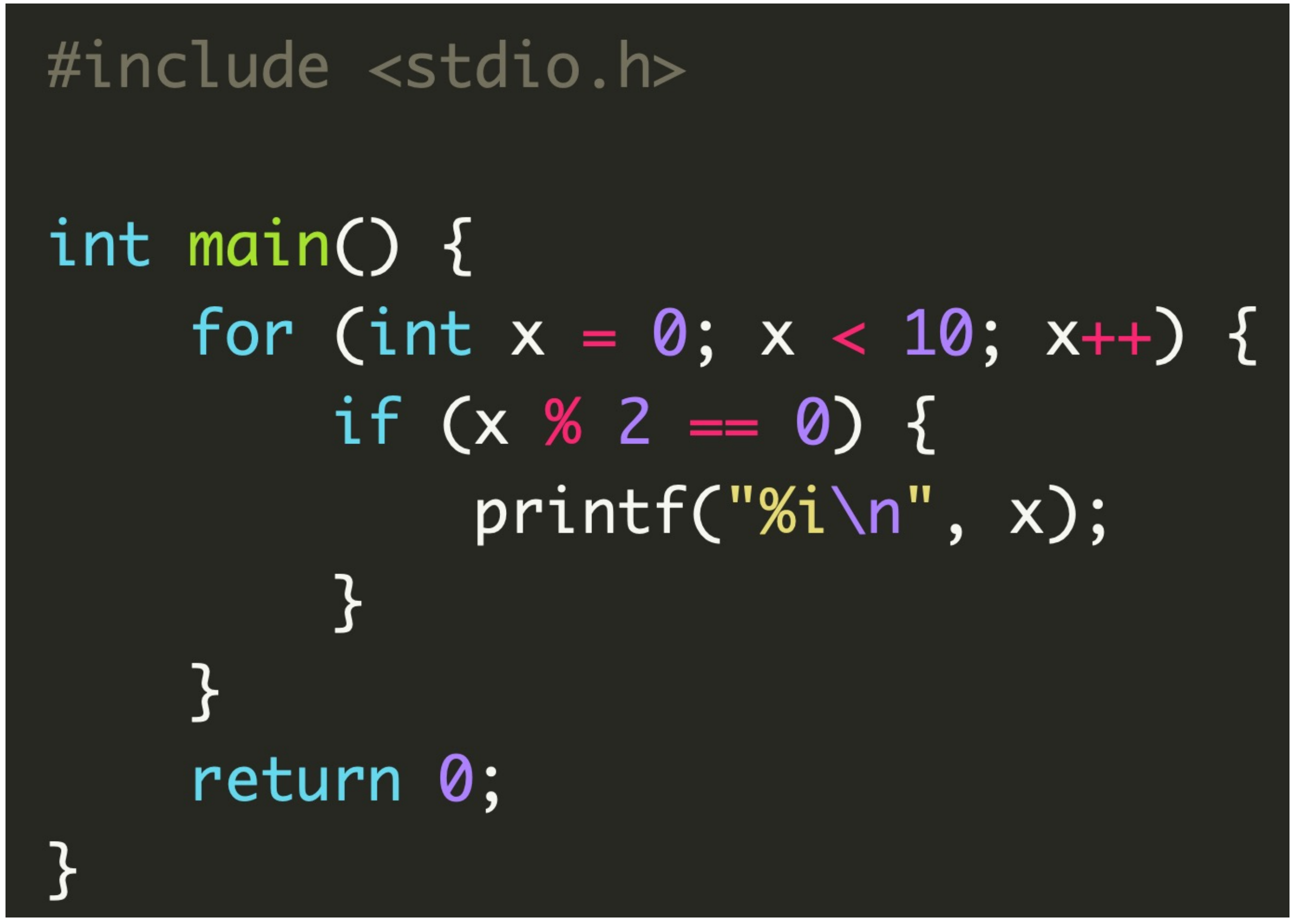
```c
#include <stdio.h>

int main() {
    for (int x = 0; x < 10; x++) {
            if (x % 2 == 0) {
                printf("%i\n", x);
        }
}
return 0;
}
```

Spacing

- To make it easier to read a statement you should have one space between variables, operators, and literals.

Spacing

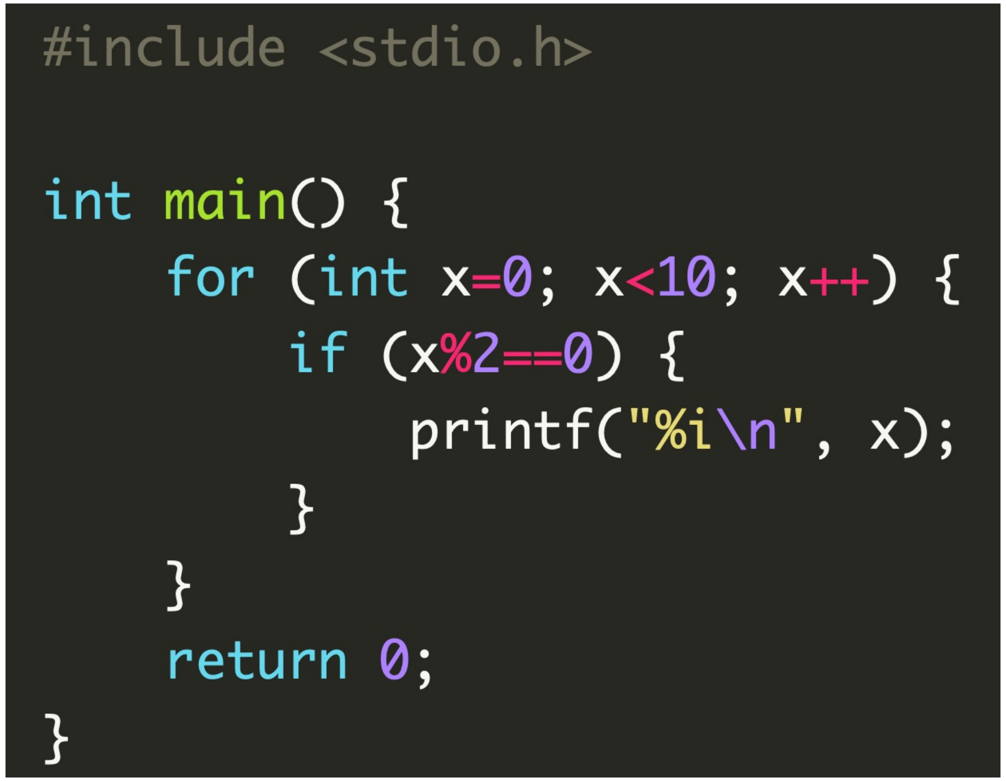
- DO: Proper spacing inside the statement



```c
#include <stdio.h>

int main() {
    for (int x = 0; x < 10; x++) {
        if (x % 2 == 0) {
            printf("%i\n", x);
        }
    }
    return 0;
}
```

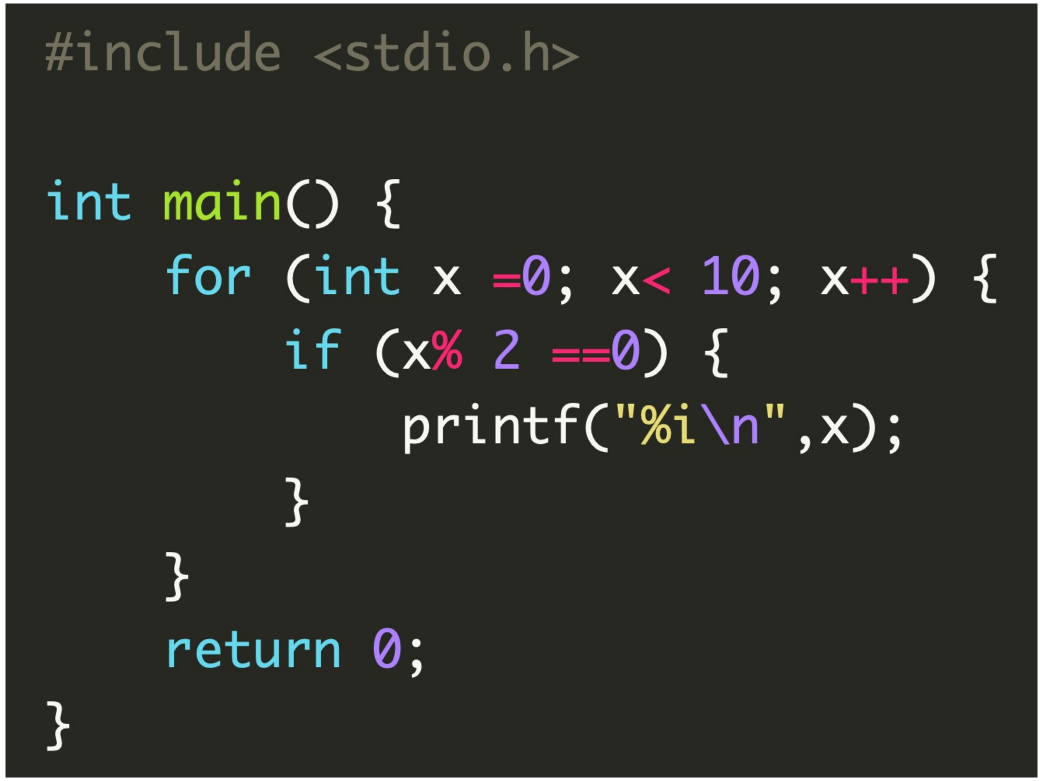
DON’T

Statements with missing spacing



```c
#include <stdio.h>

int main() {
    for (int x=0; x<10; x++) {
        if (x%2==0) {
            printf("%i\n", x);
        }
    }
    return 0;
}
```

Use inconsistent spacing

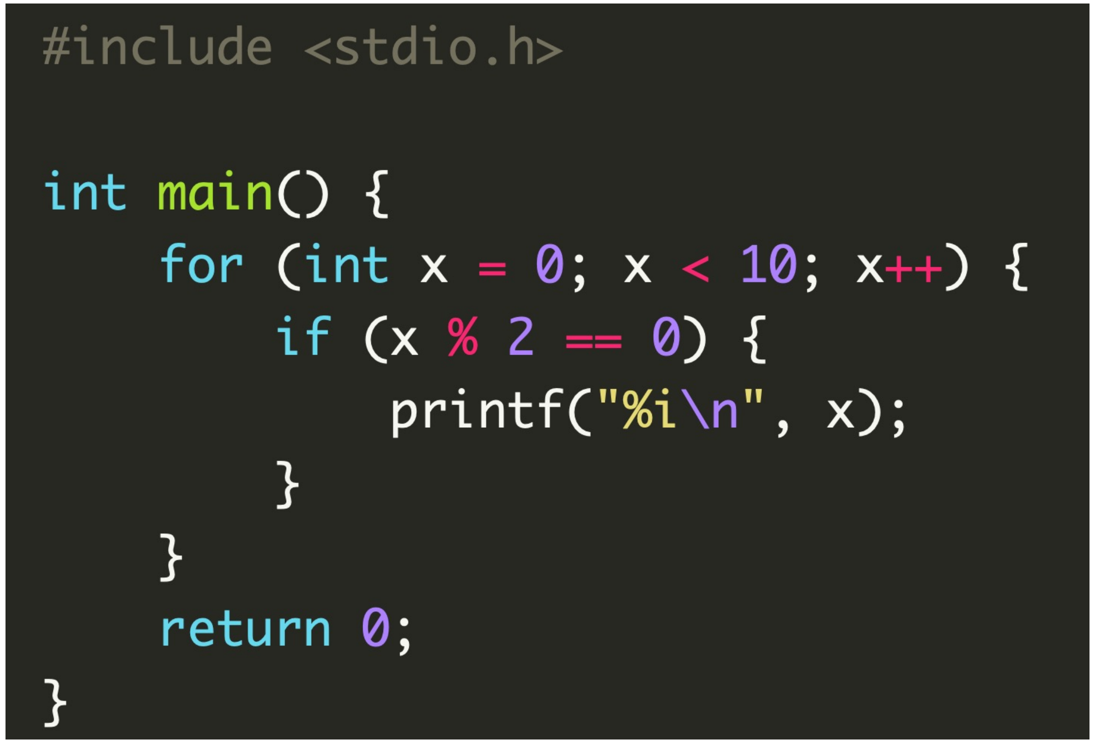


```c
#include <stdio.h>

int main() {
    for (int x =0; x< 10; x++) {
        if (x% 2 ==0) {
            printf("%i\n",x);
        }
    }
    return 0;
}
```

Blocks

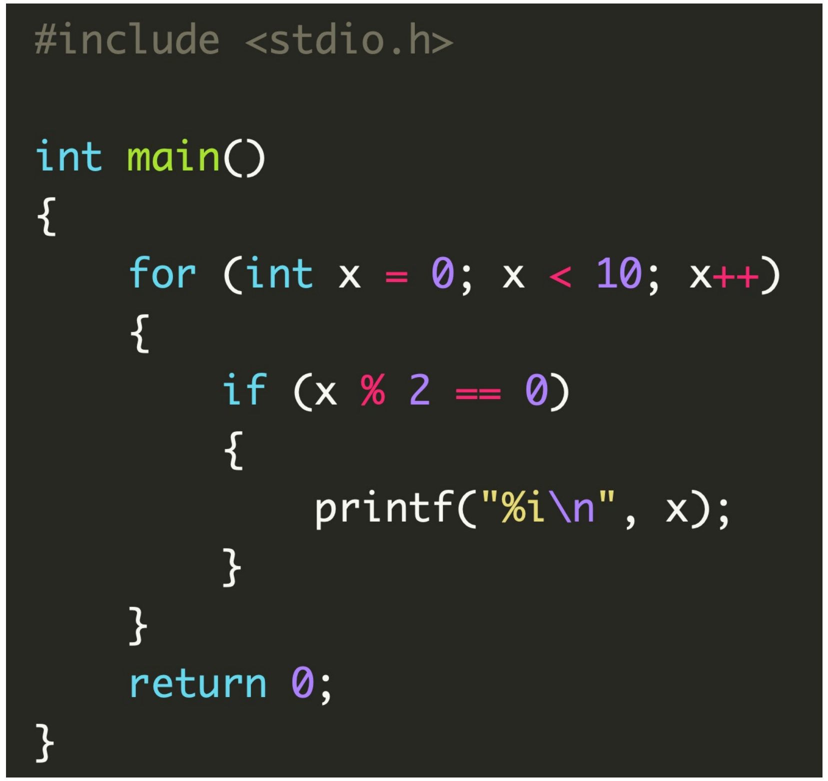
- Curly braces are used to indicate the start and end of functions, loops, conditional, and blocks.

DO: Start curly braces on the same line a block begins



```c
#include <stdio.h>

int main() {
    for (int x = 0; x < 10; x++) {
        if (x % 2 == 0) {
            printf("%i\n", x);
        }
    }
    return 0;
}
```

DON'T: Start your curly braces on the next line



```c
#include <stdio.h>

int main()
{
    for (int x = 0; x < 10; x++)
    {
        if (x % 2 == 0)
        {
            printf("%i\n", x);
        }
    }
    return 0;
}
```

Functions

- Function names must be lower case and use underscore between words: my_function
- Function names must reflect the purpose of the function
- Functions should have comments above describing what they do and the inputs and outputs
- All functions must have prototypes
  - in header files for multiple file programs
  - above the main function for single file programs
- Function implementations go:
  - in .c files for multiple file programs
  - below the main function for single file programs
- If you have an array as a function parameter and will not be modifying its contents you **MUST USE const**

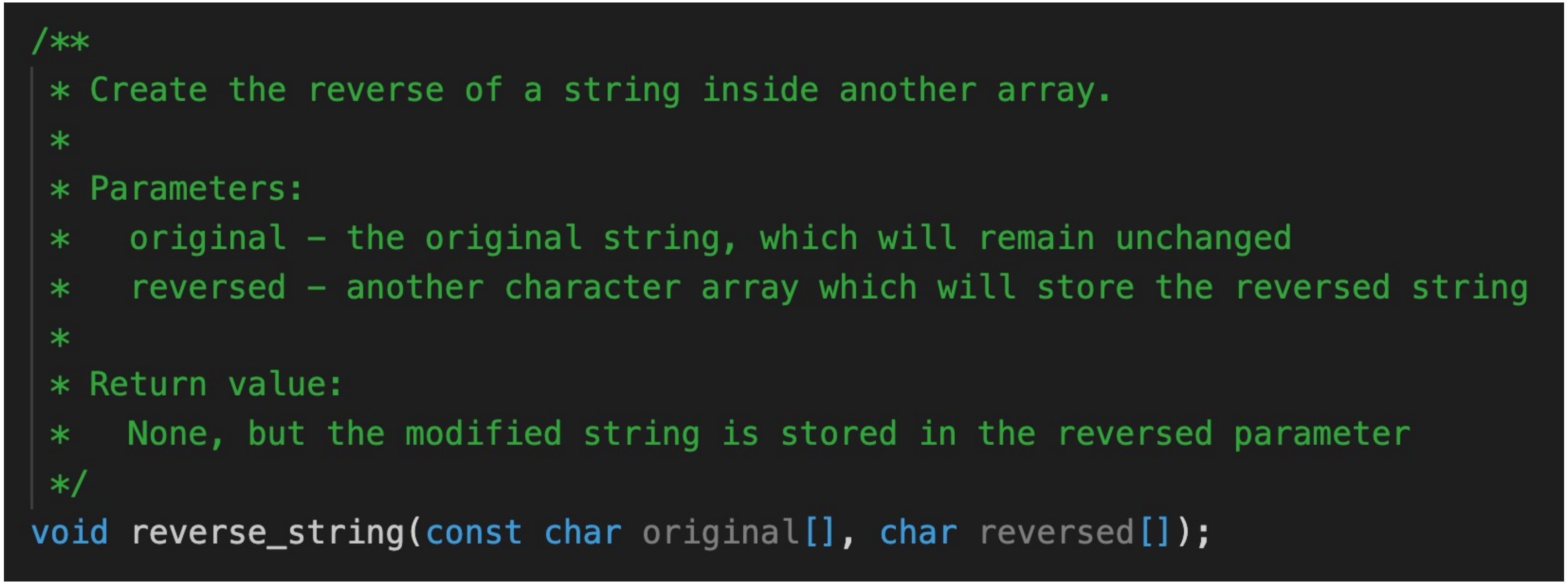
Header files

- Always have include guards

Comments

- Function prototypes should have comments above them explaining:
- The purpose of the function
- Paramaters and preconditions (if any)
- Return data and postconditions (if any)

Comments



```c
/**
 * Create the reverse of a string inside another array.
 *
 * Parameters:
 *   original - the original string, which will remain unchanged
 *   reversed - another character array which will store the reversed string
 *
 * Return value:
 *   None, but the modified string is stored in the reversed parameter
 */
void reverse_string(const char original[], char reversed[]);
```

Naming Rules

- Variable names should be lower case and use underscore between words: my_var
- Constant variables and Macro names are in all caps and use underscore between words: MY_CONST_VAR
- Variable names must be descriptive
  - Single-letter variable names are only allowed if it is immediately obvious what they mean.
    - Using i, j, and k as loop variables is common
    - some problems have an abstract quantity n or x, but otherwise choose a more descriptive name.
  - Multiple letter abbreviations are ok as long as the meaning is obvious (like col for column) but often writing out the entire word is clearer.
  - Use plural nouns for collections like lists and arrays. If a variable stores a quantity, like the number of letters in a word, do not use letters but instead use letter_count.
  - Clarity is better than brevity. monthy_pay is better than mon_pay or simply pay