Handy functions to work with strings

string.h is a C library which has many functions to work with strings in C

There are helper methods to deal with getting the length of a string, concatenate 2 strings, copy from one string to another etc

It is great pointer practice to implement these string functions - let’s work with a few of those

A function to find the length of a given string

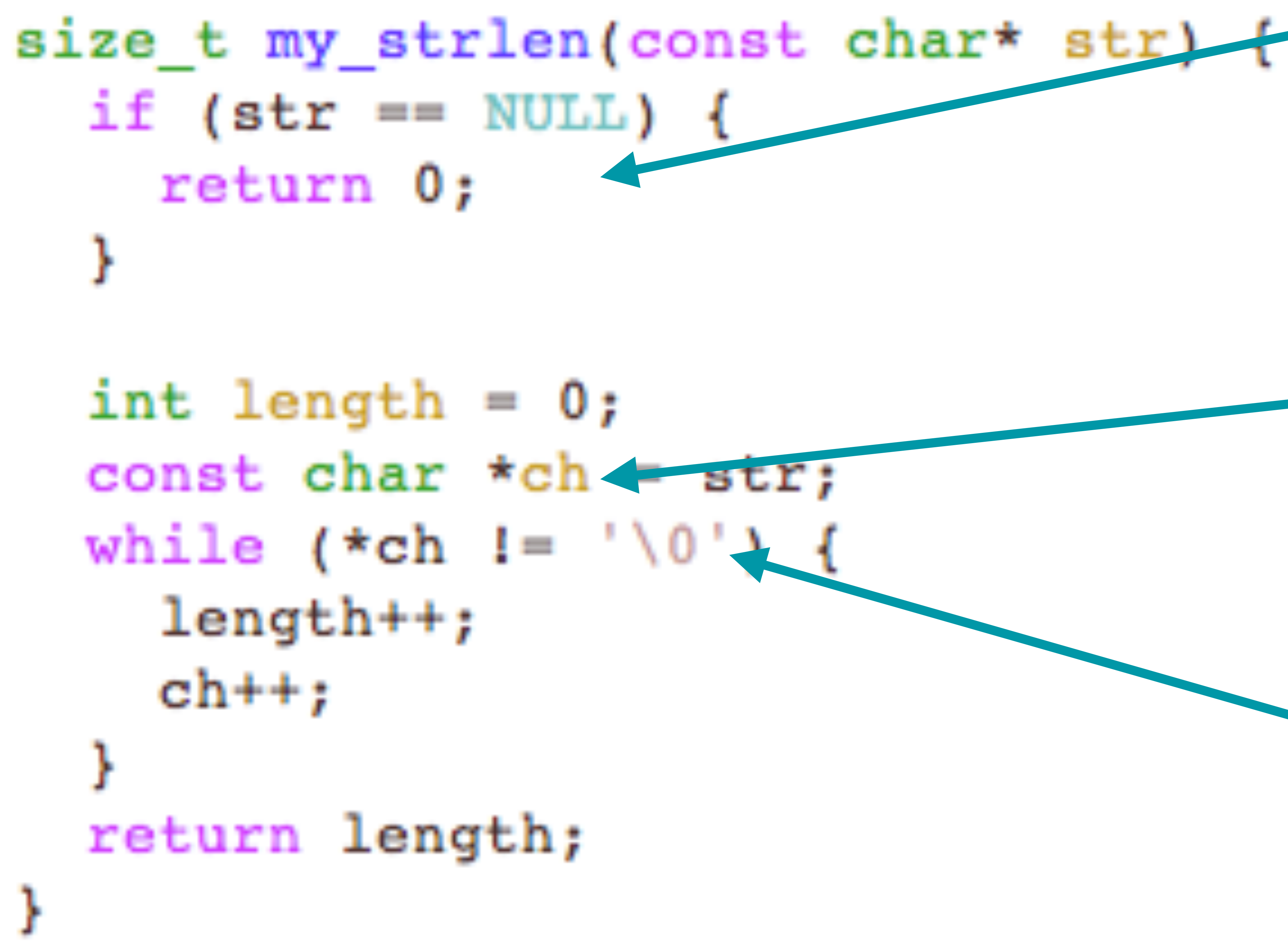
```
size_t strlen(const char* str)
```

size_t: is an unsigned integral type, the length of a string cannot be negative so this makes sense

const char* size: this indicates that the pointer is to a constant char which means the string it is pointing to cannot be changed, operations such as `str[0] = ‘p’` is not valid

Implement your own `size_t my_strlen(const char* str)`. Makes sure that it works for all string lengths and handles errors correctly

STRING LENGTH IMPLEMENTATION



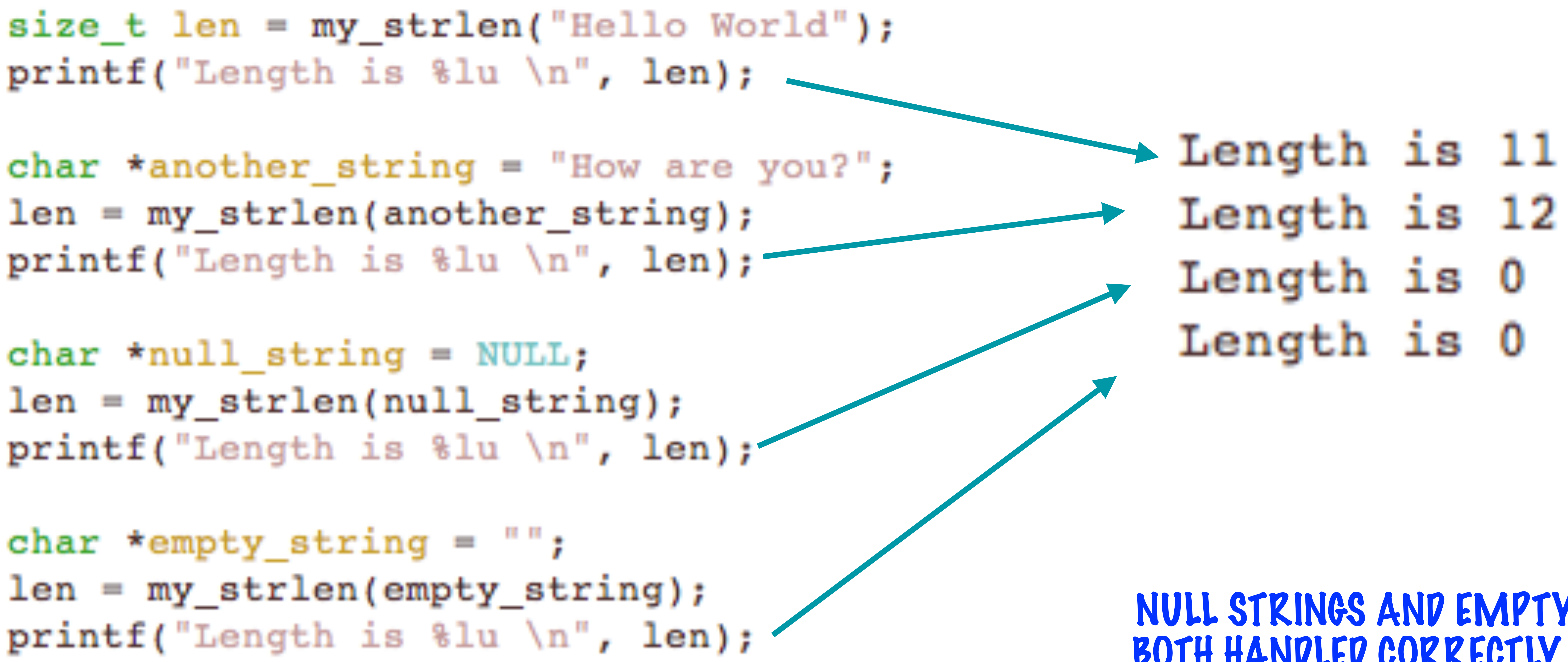
```
size_t my_strlen(const char* str) {
  if (str == NULL) {
    return 0;
  }

  int length = 0;
  const char *ch = str;
  while (*ch != '\0') {
    length++;
    ch++;
  }
  return length;
}
```

THIS CHECK IS IMPORTANT, THESE DETAILS ARE IMPORTANT TO THE INTERVIEWER

WE COULD CHOOSE TO INCREMENT STR ITSELF BUT IT SEEMS CLEANER TO USE ANOTHER VARIABLE

REMEMBER ALL STRINGS IN C ARE TERMINATED BY '\0'

my_strlen WORKS WITH ALL STRINGS AND STRING LENGTHS



```c
size_t len = my_strlen("Hello World");
printf("Length is %lu \n", len);

char *another_string = "How are you?";
len = my_strlen(another_string);
printf("Length is %lu \n", len);

char *null_string = NULL;
len = my_strlen(null_string);
printf("Length is %lu \n", len);

char *empty_string = "";
len = my_strlen(empty_string);
printf("Length is %lu \n", len);
```

```
Length is 11
Length is 12
Length is 0
Length is 0
```

NULL STRINGS AND EMPTY STRINGS ARE BOTH HANDLED CORRECTLY, THESE DETAILS ARE IMPORTANT!

A function to check for a character in a string

```
char* strchr(const char* str, int c)
```

Implement your own `char* my_strchr(const char* str, int c)`

This returns a pointer to where the character `c` is present in the string `str`

Couple of things to note:

- characters are integers at heart, int c is just the ASCII code for the character and can be tested for equality with characters
- the functions should handles nulls and return a null if the character is not found in the string

STRING CHARACTER CHECK IMPLEMENTATION

```
char* my_strchr(const char* str, int c) {
  if (str == NULL) {
    return NULL;
  }

  while (*str != '\0') {
    if (*str == c) {
      return (char*) str;
    }
    str++;
  }

  return NULL;
}
```

IF THE INPUT STRING ITSELF IS NULL THEN THE CHARACTER CANNOT BE PRESENT IN THE STRING, WE RETURN NULL

CHECKING FOR EQUALITY BETWEEN A CHARACTER AND AN INTEGER MAKES SENSE BECAUSE A CHARACTER IS REPRESENTED BY ITS ASCII CODE

WE CAST A CONST CHAR* TO A CHAR* TO SATISFY THE RETURN VALUE WITHOUT THE COMPILER GIVING US WARNINGS ABOUT THE IMPLICIT CAST TO CHAR*