Strings

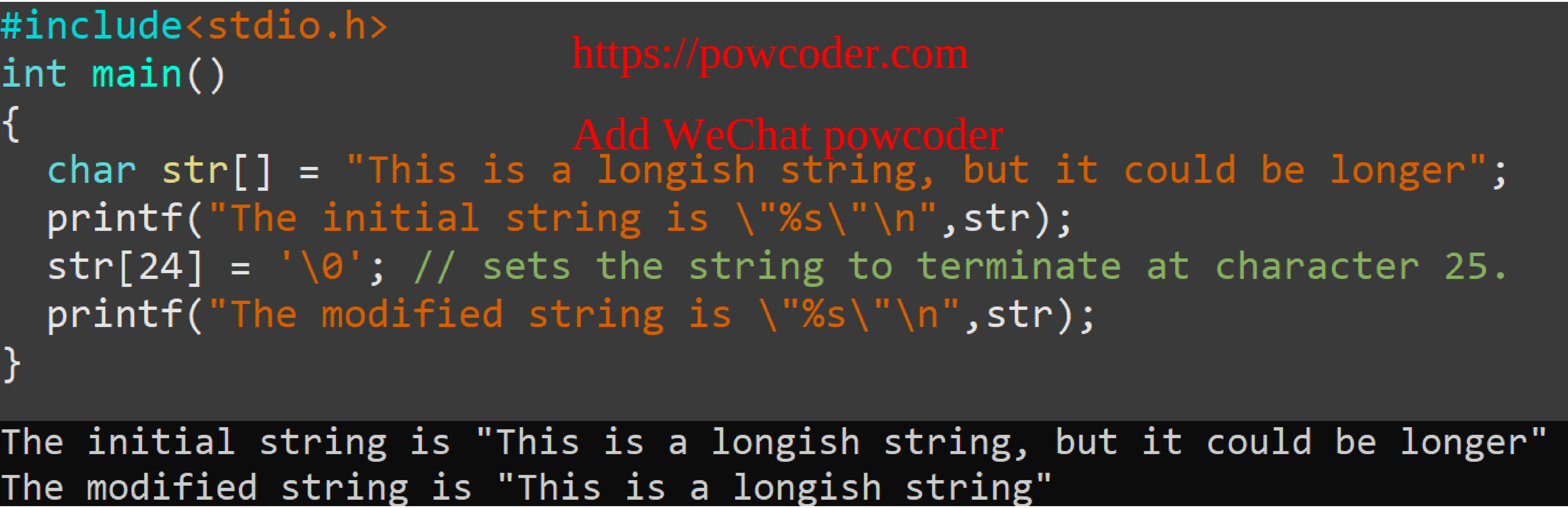
String Literals vs Character Arrays

- C has three main ways to declare strings.

- String literal.
  - Creates a pointer that points to a string in read-only memory.
  - Cannot be changed but easy to declare and use.
- Character array
  - Creates a pointer that point to a block of memory on stack frame.
  - Can be changed but not resized
- Dynamically allocated strings
  - Creates a pointer to a block of memory on the heap.
  - Can be resized, changed but needs to de-allocated to prevent memory leaks.

Internal representation of strings

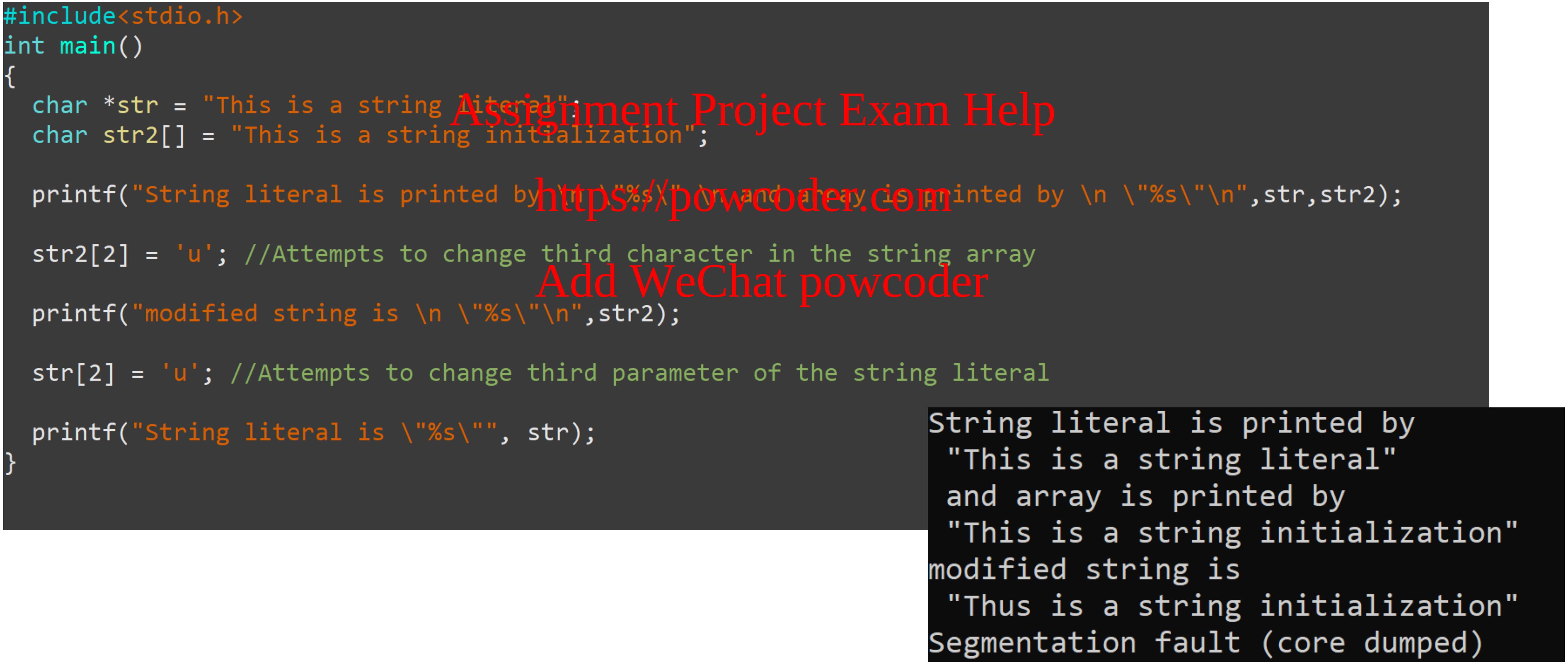
- The end of strings is specified by a special character ‘\0’
- char str[] = “This is a string”; -> “This is a string\0”;



```c
#include<stdio.h>
int main()
{
  char str[] = "This is a longish string, but it could be longer";
  printf("The initial string is \"%s\"\n",str);
  str[24] = '\0'; // sets the string to terminate at character 25.
  printf("The modified string is \"%s\"\n",str);
}
```

```
The initial string is "This is a longish string, but it could be longer"
The modified string is "This is a longish string"
```

String Literals vs Character Arrays



```
#include<stdio.h>
int main()
{
  char *str = "This is a string literal";
  char str2[] = "This is a string initialization";

  printf("String literal is printed by \n \"%s\" \n and array is printed by \n \"%s\"\n",str,str2);

  str2[2] = 'u'; //Attempts to change third character in the string array

  printf("modified string is \n \"%s\"\n",str2);

  str[2] = 'u'; //Attempts to change third parameter of the string literal

  printf("String literal is \"%s\"", str);
}
```

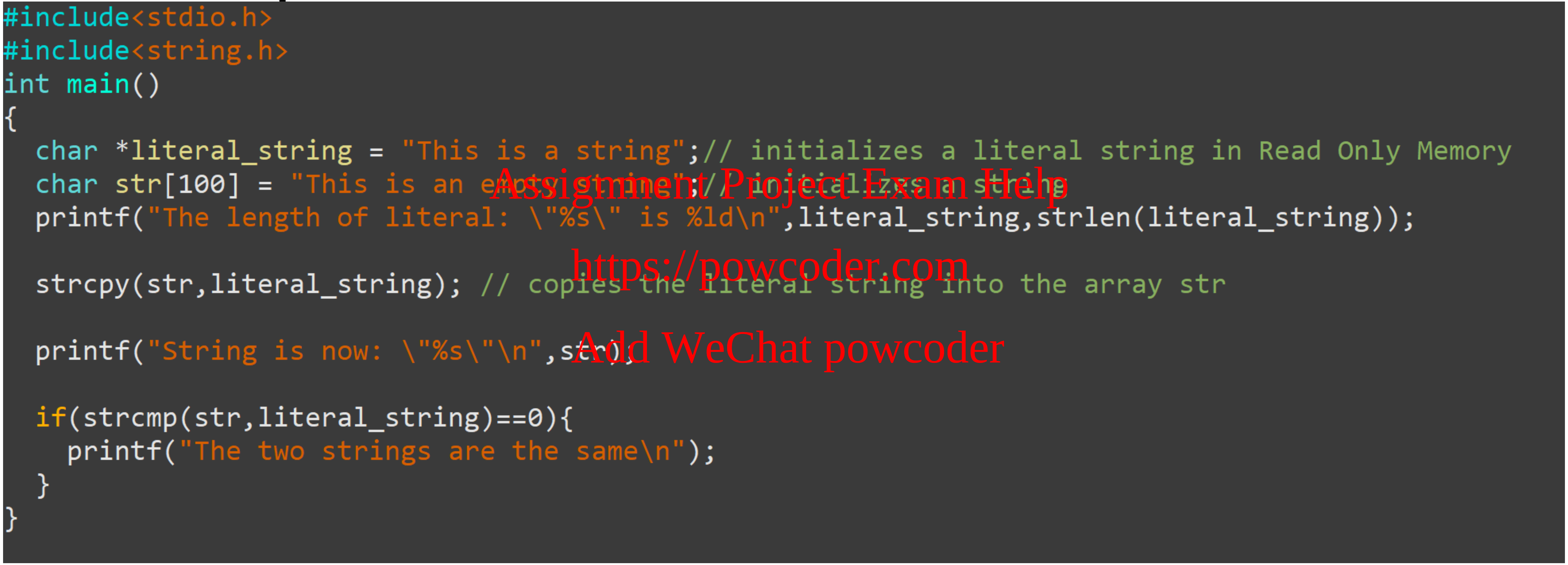
```
String literal is printed by
 "This is a string literal"
 and array is printed by
 "This is a string initialization"
modified string is
 "Thus is a string initialization"
Segmentation fault (core dumped)
```

String features in c

- String.h gives a bunch of useful functions for handling strings in c.

- Strcpy: makes a copy by value of a string
- Strcat: concatenates two string (be careful about memory!).
- Strlen: computes the length of a string
- Strcmp: compare two strings

Example

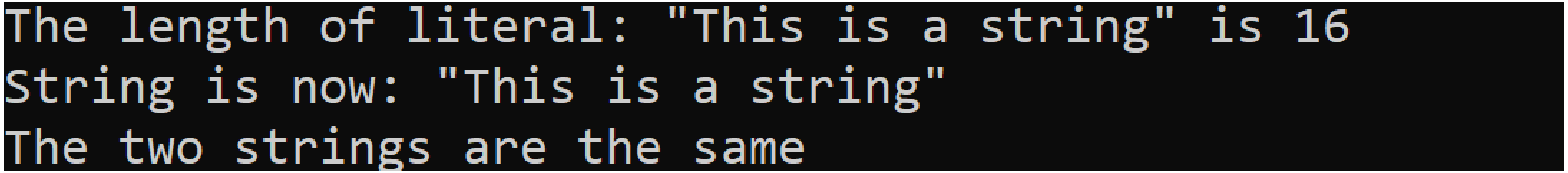


```
#include<stdio.h>
#include<string.h>
int main()
{
  char *literal_string = "This is a string";// initializes a literal string in Read Only Memory
  char str[100] = "This is an empty string";// initializes a string
  printf("The length of literal: \"%s\" is %ld\n",literal_string,strlen(literal_string));

  strcpy(str,literal_string); // copies the literal string into the array str

  printf("String is now: \"%s\"\n",str);

  if(strcmp(str,literal_string)==0){
    printf("The two strings are the same\n");
  }
}
```



```
The length of literal: "This is a string" is 16
String is now: "This is a string"
The two strings are the same
```

Arrays of strings

- Char planets[][8] ={“Mercury”, “Venus”, “Earth”, “Mars”, “Jupiter”, “Saturn”, “Uranus”, “Neptune”, “Pluto?”};

M	E	R	C	U	R	Y	\0
V	E	N	U	S	\0	\0	\0
E	A	R	T	H	\0	\0	\0
M	A	R	S	\0	\0	\0	\0
J	U	P	I	T	E	R	\0
S	A	T	U	R	N	\0	\0
U	R	A	N	U	S	\0	\0
N	E	P	T	U	N	E	\0
P	L	U	T	O	?	\0	\0

Array of string literals

- Char *planets[] = {“Mercury”, “Venus”, “Earth”, “Mars”, “Jupiter”, “Saturn”, “Uranus”, “Neptune”, “Pluto?”}