C Strings

September 23, 2019 8:03 AM

Strings

- different ways to create strings
  char an_array[6] = {'H', 'e', 'l', 'l', 'o', ' ', 'M', 'e'}
  char str[SOMESIZE] = "A string of char";
  - automatically gives a null char at the end
  char another string = "A string of char";*
- can't copy strings of different lengths:
  char str1[128] = "first string";
  char str2[200] = "second string";
  str2 = str1 OR --> WRONG XX
- String length
  - provides group of functions for string processing header: *#include <string.h>*
  - string length: *size_t strlen(const char* s);*

String comparison

- *str1 == str2* to compare the pointers --> do they occupy same space in memory?
  - does not compare the string values
- **Comparing chars**
  char a = 'a'; char b = 'b';
  if (a == b){...}
- **Compare strings in C**
  int strcmp(const char str1, const char* str2);*
  int strncmp(cons char str1, const char* str2, size_t num);*

```
char string1[] = "Hello";
char string2[] = "Hello there";
int length;

length = strlen(string1);
if (strncmp(string1, string2, length) == 0) {
  printf("The first %d letters of %s and
          %s are the same\n", length, string1, string2);
} else {
  printf("Not the same\n");
}
```

- **String searching**
  - check if string contains another string
    char strstr(const char* hastack, const char* needle);*
  - locates first occurrence in haystack of entire needle string string or NULL
    char string1[] = "feed";
    char string2[] = "Don't feed the bear!";
    result = strstr(string2, string1);
    returns: "feed the bear!"
  - if want to check if contains "feed" again later on, check for later occurrences by checking after the word:
    strstr(result, string1)
    - *result* is *strlen(string1)*
    - if allowed to overlap, check from second letter instead of *strlen*
- **String concatenation**
  int strncat(char sq, const char* s2, size_t n);*
  - appends no more than n bytes from s2 to the end of s1
  - initial byte of s2 overwrites null byte of s1
  - terminating null byte appended to result

```
char* empty_string;
char a_long_string[128] = "These ";
strcat(a_long_string, "strings ");
strcat(a_long_string, "are ");
empty_string = strcat(a_long_string, "concatenated!");
printf("%s\n", empty_string);
```

- **String copying**
  char strncpy(char* s1, const char* s2, size_t n);*
  - copies no more than n bytes from the string pointed to by s2 to the string pointed to by s1

```
char a_str[] = "Make the news.";
int length = strlen(a_str);
char* other_str = (char*) malloc(length+1); // why +1?
strcpy(other_str, a_str);
a_str[0] = 'F';
printf("a_str = %s\notherstr = %s\n", a_str, other_str);
```