

Sprint 07

Marathon C

April 15, 2020



ucode

Contents



Engage	2
Investigate	3
Act: Task 00 > New string	5
Act: Task 01 > Duplicate string	6
Act: Task 02 > Join strings	7
Act: Task 03 > Copy array of Integers	8
Act: Task 04 > Delete string	9
Act: Task 05 > Concatenate words	10
Act: Task 06 > Trim string	11
Act: Task 07 > Clean string	12
Act: Task 08 > Split string	13
Act: Task 09 > Delete duplicates	14
Act: Task 10 > Delete array of strings	15
Share	16

Engage



DESCRIPTION

Hey!

It's been a long time since you started learning C.

It's time to explore a very important topic. Namely, working with memory. Today you will learn why it is so important and powerful. Keep in mind that the success of your future education depends on the depth of knowledge and awareness of this topic.

This **Sprint** will expand your horizons in programming. You will start to use the `malloc` and `free` functions to work with memory.

Let's begin.

BIG IDEA

Learning dynamical memory allocation.

ESSENTIAL QUESTION

How to allocate and work with memory in the C language?

CHALLENGE

Learn how to manage memory in the C language.

Act: Task 00



NAME

New string

DIRECTORY

t00/

SUBMIT

mx_strnew.c

ALLOWED FUNCTIONS

malloc

DESCRIPTION

Create a function that:

- allocates memory for a string of a specific `size` and one additional byte for the terminating `'\0'`
- initializes each character with `'\0'`

RETURN

- returns the string of a specific `size` and terminated by `'\0'`
- returns `NULL` if creation fails

SYNOPSIS

```
char *mx_strnew(const int size);
```

EXAMPLE

```
mx_strnew(10); //returns string with size 10 and terminated by '\0'  
mx_strnew(-1); //returns NULL
```

FOLLOW THE WHITE RABBIT

man 3 malloc

Act: Task 01



NAME

Duplicate string

DIRECTORY

```
t01/
```

SUBMIT

```
mx_strdup.c, mx_strnew.c, mx_strlen.c, mx_strcpy.c
```

ALLOWED FUNCTIONS

```
malloc
```

DESCRIPTION

Create a function that has the same behaviour as the standard libc function `strdup`.

SYNOPSIS

```
char *mx_strdup(const char *str);
```

FOLLOW THE WHITE RABBIT

```
man 3 strdup
```

Act: Task 02

NAME

Join strings

DIRECTORY

t02/

SUBMIT

mx_strjoin.c, mx_strnew.c, mx_strlen.c, mx_strdup.c, mx_strcpy.c, mx_strcat.c

ALLOWED FUNCTIONS

malloc

DESCRIPTION

Create a function that:

- concatenates strings `s1` and `s2` into a new string
- terminates the new string with `'\0'`

RETURN

- returns the string as a result of concatenation `s1` and `s2`
- returns the new copy of `non-NULL` parameter if one and only one of the parameters is `NULL`
- returns `NULL` if the concatenation fails

SYNOPSIS

```
char *mx_strjoin(char const *s1, char const *s2);
```

EXAMPLE

```
str1 = "this";
str2 = "dodge ";
str3 = NULL;
mx_strjoin(str2, str1); //returns "dodge this"
mx_strjoin(str1, str3); //returns "this"
mx_strjoin(str3, str3); //returns NULL
```


Act: Task 03



NAME

Copy array of integers

DIRECTORY

```
t03/
```

SUBMIT

```
mx_copy_int_arr.c
```

ALLOWED FUNCTIONS

```
malloc
```

DESCRIPTION

Create a function that copies an array of integers to a new array.

RETURN

- returns the `pointer` to the first element
- returns `NULL` if the array `src` does not exist or copying fails

SYNOPSIS

```
int *mx_copy_int_arr(const int *src, int size);
```

EXAMPLE

```
arr1 = {1, 2, 3};  
arr2 = NULL;  
mx_copy_int_arr(arr1, 3); //returns array [1, 2, 3]  
mx_copy_int_arr(arr2, 3); //returns NULL
```

Act: Task 04



NAME

Delete string

DIRECTORY

t04/

SUBMIT

mx_strdel.c

ALLOWED FUNCTIONS

free

DESCRIPTION

Create a function that:

- takes a pointer to a string
- frees the string's memory with `free`
- sets the string to `NULL`

SYNOPSIS

```
void mx_strdel(char **str);
```

FOLLOW THE WHITE RABBIT

man 3 malloc

Act: Task 05



NAME

Concatenate words

DIRECTORY

```
t05/
```

SUBMIT

```
mx_concat_words.c, mx_strdel.c, mx_strjoin.c, mx_strnew.c, mx_strlen.c, mx_strdup.c,  
mx_strcpy.c, mx_strcat.c
```

ALLOWED FUNCTIONS

```
malloc, free
```

DESCRIPTION

Create a function that:

- concatenates the `NULL`-terminated array of words into a sentence where words are separated by a single space character
- frees all unused memory

RETURN

- returns the result of concatenation of the `NULL`-terminated array into a string
- returns `NULL` if the array of strings `words` does not exist or concatenation fails

SYNOPSIS

```
char *mx_concat_words(char **words);
```

EXAMPLE

```
words = {"Free", "your", "mind.", NULL};  
mx_concat_words(words); //returns "Free your mind."  
mx_concat_words(NULL); //returns NULL
```

SEE ALSO

[Memory leaks](#)

Act: Task 06



NAME

Trim string

DIRECTORY

t06/

SUBMIT

`mx_strtrim.c, mx_strdel.c, mx_isspace.c, mx_strnew.c, mx_strlen.c, mx_strncpy.c`

ALLOWED FUNCTIONS

`malloc, free`

DESCRIPTION

Create a function that:

- creates a new string without whitespace characters at the beginning and the end of the string
- frees all unused memory

RETURN

- returns a new trimmed string
- returns `NULL` if the string `str` does not exist or string trim fails

SYNOPSIS

```
char *mx_strtrim(const char *str);
```

EXAMPLE

```
name = "\f My name... is Neo \t\n ";  
mx_strtrim(name); //returns "My name... is Neo"
```

Act: Task 07

NAME

Clean string

DIRECTORY

t07/

SUBMIT

```
mx_del_extra_whitespace.c, mx_strtrim.c, mx_isspace.c, mx_strncpy.c, mx_strnew.c,  
mx_strdel.c, mx_strlen.c
```

ALLOWED FUNCTIONS

malloc, free

DESCRIPTION

Create a function that:

- creates a new string without whitespace characters in the beginning and/or at the end of the string
- separates words in the new string with exactly one space character
- frees all unused memory

A word is a sequence of characters separated by whitespaces.

RETURN

- returns a new created string
- returns `NULL` if the string `str` does not exist or string creation fails

SYNOPSIS

```
char *mx_del_extra_whitespace(const char *str);
```

EXAMPLE

```
name = "\f My name...    is \r Neo \t\n ";  
mx_del_extra_whitespace(name); //returns "My name... is Neo"
```

Act: Task 08

NAME

Split string

DIRECTORY

t08/

SUBMIT

mx_strsplit.c, mx_strnew.c, mx_strncpy.c, mx_strdel.c, mx_count_words.c

ALLOWED FUNCTIONS

malloc, free

DESCRIPTION

Create a function that:

- converts a string `s` to the `NULL`-terminated array of words
- frees all unused memory

A word is a sequence of characters separated by the character `c` as a delimiter.

RETURN

- returns the `NULL`-terminated array of strings
- returns `NULL` if the string `s` does not exist or conversion fails

SYNOPSIS

```
char **mx_strsplit(char const *s, char c);
```

EXAMPLE

```
s = "**Good bye,**Mr.*Anderson.**";
arr = mx_strsplit(s, '*'); // arr = ["Good bye,", "Mr.", "Anderson."]
s = "    Knock, knock,    Neo.    ";
arr = mx_strsplit(s, ' '); // arr = ["Knock,", "knock,", "Neo."]
```

Act: Task 09



NAME

Delete duplicates

DIRECTORY

t09/

SUBMIT

mx_del_dup_arr.c, mx_copy_int_arr.c

ALLOWED FUNCTIONS

malloc

DESCRIPTION

Create a function that:

- takes an array of integers `src`, its size `src_size` and the pointer to the size of the new array `dst_size`
- initializes `dst_size` by the size of the array without duplicates
- creates the new array without duplicates

RETURN

- returns a new array without duplicates
- returns `NULL` if the array `src` does not exist or creation fails

SYNOPSIS

```
int *mx_del_dup_arr(int *src, int src_size, int *dst_size);
```

Act: Task 10



NAME

Delete array of strings

DIRECTORY

t10/

SUBMIT

mx_del_strarr.c, mx_strdel.c

ALLOWED FUNCTIONS

free

DESCRIPTION

Create a function that:

- takes a pointer to the `NULL`-terminated array of strings
- deletes the content of an array
- frees the array memory with `free`
- sets the pointer to `NULL`

SYNOPSIS

```
void mx_del_strarr(char ***arr);
```