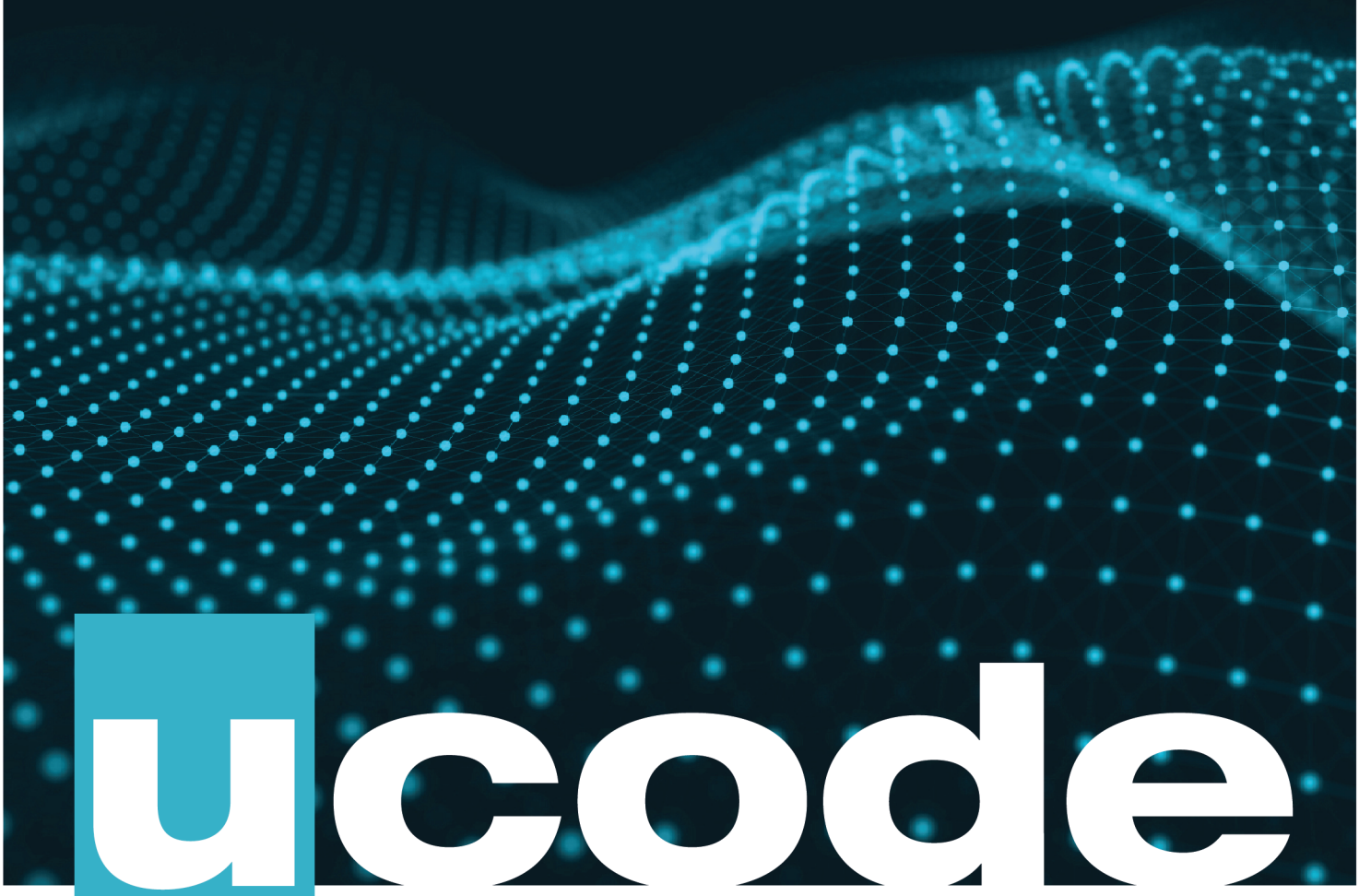
# Sprint 03

Marathon C

October 15, 2019



ucode

# Contents



ucode

# Engage

## DESCRIPTION

Hi.

Well going. Let's move on, complicating our challenges.

During this sprint, you'll learn pointers in C and write more complex algorithms. This is a very important basic construction. Therefore, treat this challenge with special attention.

## BIG IDEA

Learn to constantly learn.

## ESSENTIAL QUESTION

What knowledge is important to you now?

## CHALLENGE

Learn to use pointers in C.

# Investigate

## GUIDING QUESTIONS

We invite you to find answers to the following questions. This will help you realize what knowledge you will get from this challenge and how to move forward.

Ask your neighbor on the right, left, or behind you, and discuss the following questions together. You can find the answers in the Internet and share it with student around you.

We encourage you to ask as many questions about `C` programming as possible. Note down your discussion.

- What are the advantages of ucode learning system?
- How many people did you communicate and work with yesterday? 4, 8, 15, 16, 23..?
- What are your impressions of the assessments? Reflection?
- What did you learn during the assessment of another student?
- What is the biggest discovery in `C` for you at the moment?
- What is still unclear in `C` for you at this time?
- How to transform uppercase to lowercase?
- What is `write` function? What do you know about it?
- What are `pointers`? Are there `strings` in C?

## GUIDING ACTIVITIES

These are only a set of example activities and resources. Do not forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

1. Repeat the basics from yesterday. Write a program that will output integer values on the standard output using C (`mx_printint.c`), if you didn't do it yesterday.
2. Spend time to fill the gaps in knowledge from previous sprints.
3. If you have questions, don't be ashamed to ask them to other students. Peer-to-peer is your key to success.
4. Take the most difficult task from previous sprints, which you could not do before and do it. You can everything! :)

## ANALYSIS

You need to analyze all the collected information before you start.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Perform only those tasks that are given in the story.
- You should submit only the specified files in the required directory and nothing else. In case you are allowed to submit any files you should submit only files that you used to complete a task. Garbage shall not pass.
- You should compile C files with clang compiler and use these flags: `-std=c11 -Wall -Wextra -Werror -Wpedantic`.
- You should use only functions which allowed in a certain task.
- Usage of forbidden functions is considered as cheat and your challenge will be failed.
- You must complete tasks according to the rules specified in `the Auditor`.
- Your exercises will be checked and graded by students. The same as you. `Peer-to-Peer (P2P) learning`.
- Also, your exercises will pass automatic evaluation which is called `Oracle`.
- Got a question or you do not understand something? Ask the students or just Google that.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!!!

# Act

## SOLUTION DEVELOPMENT

Let's get started! And may the odds be ever in your Favor!

1. Clone your git repository, what is issued on the challenge page in the LMS.
2. Open the story and read it!
3. Arrange to brainstorm tasks with other students.
4. Try to realize your thoughts in code.

# Task 00

## NAME

Dereferencing a pointer

## DIRECTORY

`t00/`

## SUBMIT

`mx_deref_pointer.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will take as parameter `******str` pointer to pointer to pointer to pointer to pointer to pointer of `char` and sets `Follow the white rabbit!` to the pointer of `char`.

## SYNOPSIS

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

## SEE ALSO

Pointers in C

# Task 01

## NAME

Referencing a pointer

## DIRECTORY

`t01/`

## SUBMIT

`mx_ref_pointer.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will take as parameter `int i` and sets its value to another parameter `int ******ptr` which is a pointer to pointer to pointer to pointer to pointer to the pointer of `int`.

## SYNOPSIS

```
void mx_ref_pointer(int i, int ******ptr);
```

## SEE ALSO

Pointers in C

# Task 02

## NAME

Reverse case

## DIRECTORY

`t02/`

## SUBMIT

`mx_reverse_case.c, mx_tolower.c, mx_toupper.c, mx_islower.c, mx_isupper.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will reverse string characters case in place.

## SYNOPSIS

```
void mx_reverse_case(char *s);
```

## EXAMPLE

```
str = "HeLLO Neo!";
mx_reverse_case(str); //converts to "hEllo nEO!"
```

# Task 03

## NAME

Swap characters

## DIRECTORY

`t03/`

## SUBMIT

`mx_swap_char.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will swap the characters of the string using pointers.

## SYNOPSIS

```
void mx_swap_char(char *s1, char *s2);
```

## EXAMPLE

```
str = "ONE";
mx_swap_char(&str[0], &str[1]); //'str' now is "NOE"
mx_swap_char(&str[1], &str[2]); //'str' now is "NEO"
```

# Task 04

## NAME

Reverse string

## DIRECTORY

`t04/`

## SUBMIT

`mx_str_reverse.c, mx_strlen.c, mx_swap_char.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will reverse string using pointers.

## SYNOPSIS

```
void mx_str_reverse(char *s);
```

## EXAMPLE

```
str = "game over";
mx_str_reverse(str); //'str' now is "revo emag"
```

# Task 05

## NAME

Compare strings

## DIRECTORY

`t05/`

## SUBMIT

`mx_strcmp.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which has the same behaviour as standard libc function `strcmp`.

## SYNOPSIS

```
int mx_strcmp(const char *s1, const char *s2);
```

## FOLLOW THE WHITE RABBIT

man 3 strcmp

# Task 06

## NAME

Copy string

## DIRECTORY

`t06/`

## SUBMIT

`mx_strcpy.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which has the same behaviour as standard libc function `strcpy`.

## SYNOPSIS

```
char *mx_strcpy(char *dst, const char *src);
```

## FOLLOW THE WHITE RABBIT

man 3 strcpy

# Task 07

## NAME

Separate string

## DIRECTORY

`t07/`

## SUBMIT

`mx_str_separate.c, mx_printchar.c`

## ALLOWED FUNCTIONS

`write`

## DESCRIPTION

Create a function which will:

- separate given string by delimiter;
- print it on the standard output;
- each fraction must be followed by the newline.

## SYNOPSIS

```
void mx_str_separate(const char *str, char delim);
```

## CONSOLE OUTPUT

```
>./mx_str_separate | cat -e    # str = "game over", delim = ' '
game$
over$
>./mx_str_separate | cat -e    # str = "game over", delim = 'm'
ga$
e over$
>
./mx_str_separate | cat -e    # str = "MMMMatrix", delim = "M"
$
atrix$
```

# Task 08

## NAME

Exponentiation

## DIRECTORY

`t08/`

## SUBMIT

`mx_pow.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will compute `n` raised to the power of zero or positive integer `pow`.

## RETURN

Returns the result of `pow` times multiplying the number `n` by itself.

## SYNOPSIS

```
double mx_pow(double n, unsigned int pow);
```

## EXAMPLE

```
mx_pow(3, 3); //returns 27
mx_pow(2.5, 3); //returns 15.625
mx_pow(2, 0); //returns 1
```

## FOLLOW THE WHITE RABBIT

```
man pow
```

## SEE ALSO

Exponentiation

# Task 09

## NAME

Narcissistic number

## DIRECTORY

`t09/`

## SUBMIT

`mx_is_narcissistic.c, mx_pow.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will check whether number is narcissistic.

## RETURN

Returns `true` if number is narcissistic, else `false`.

## SYNOPSIS

```
bool mx_is_narcissistic(int num);
```

## EXAMPLE

```
mx_is_narcissistic(3); //returns true
mx_is_narcissistic(-3); //returns false
mx_is_narcissistic(10); //returns false
```

## SEE ALSO

Narcissistic number

# Task 10

## NAME

Prime number

## DIRECTORY

`t10/`

## SUBMIT

`mx_is_prime.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will check whether number is a prime.

## RETURN

Returns `true` if number is prime, else `false`.

## SYNOPSIS

```
bool mx_is_prime(int num);
```

## EXAMPLE

```
mx_is_prime(3); //returns true
mx_is_prime(4); //returns false
```

## SEE ALSO

Prime number

# Task 11

## NAME

Mersenne prime

## DIRECTORY

`t11/`

## SUBMIT

`mx_is_mersenne.c, mx_pow.c, mx_is_prime.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function which will check whether number is a Mersenne prime.
`Hardcoding is forbidden!`

## RETURN

Returns `true` if number is a Mersenne prime, else `false`.

## SYNOPSIS

```
bool mx_is_mersenne(int n);
```

## EXAMPLE

```
mx_is_mersenne(3); //returns true
mx_is_mersenne(11); //returns false
```

## SEE ALSO

Mersenne prime number

# Share

## PUBLISHING

The final important and integral stage of your work is its publishing. This allows you to share your challenges, solutions, and reflections with a local and global audience.

During this stage, you will find how to get a global assessment. You will get representative feedback. As a result, you get the maximum experience from the work you have done.

### What you can create to disseminate information

- Text post, summary from reflection.
- Charts, infographics or any other ways to visualize your information.
- Video of your work, reflection video.
- Audio podcast. You can record a story with your experience.
- Photos from ucode with small post.

### Example techniques

- Canva - a good way to visualize your data.
- QuickTime - easy way to record your screen, capture video, or record audio.

### Example ways to share your experience

- Facebook - create a post that will inspire your friends.
- YouTube - upload a video.
- GitHub - share your solution.
- Telegraph - create a post. This is a good way to share information in a Telegram.
- Instagram - share a photos and stories from ucode. Don't forget to tag us :)

Share what you learned with your local community and the world. Use #ucode and #CBLWorld on social media.