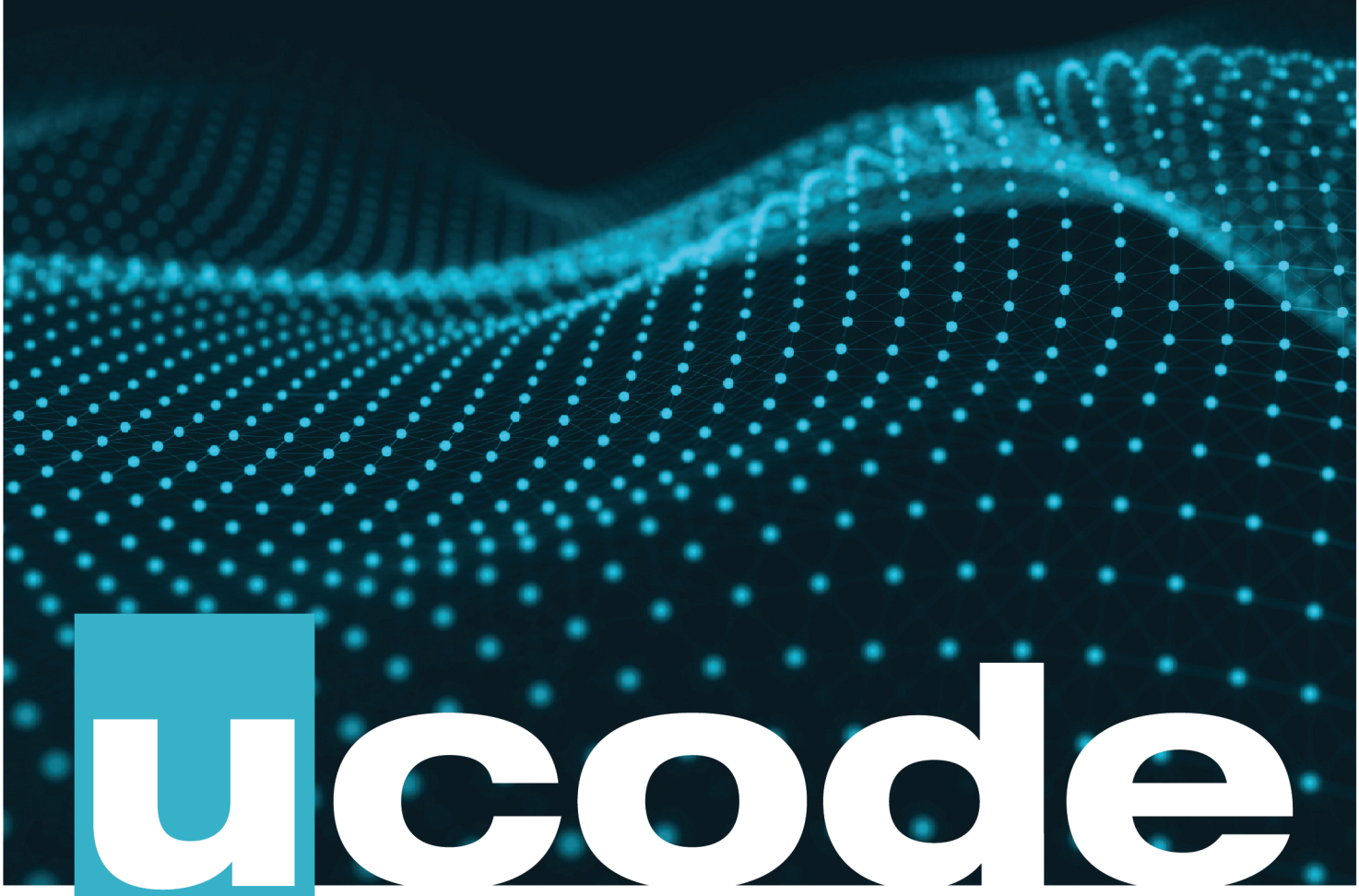
Sprint 01

Marathon C

October 22, 2019



ucode

Contents

ucode

Engage

DESCRIPTION

Hey, wazzup?

You started studying programming. That's nice! Let's go further.
During this challenge, you will learn the basics of writing a code in `C`.

We are inviting you to start learning programming from `C` because:

- `C` is the great foundation for learning other programming languages.
- `C` built on basic programming concepts and it is very simple to understand how to develop programs with its help.
- While using `C`, you always aware of how your program is working under the hood. It doesn't hide anything from you. You've got the power.
- And last, but not least, writing on `C` with accordance of `the Auditor` will help you to develop a mindset of the true programmer.

WELCOME TO THE `C`

PLEASE FOLLOW ME

BIG IDEA

Develop programmer mindset.

ESSENTIAL QUESTION

What are the components of the simplest `C` program?

CHALLENGE

Learn the basics of 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 do you remember most of yesterday?
- How is your sprint yesterday? How much tasks have you done?
- What topics were unclear to you?
- What programming languages do you know? What do you know about them?
- What do you know about `C` language?
- How will "Hello World!" 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. Discuss a weather with your neighbors. What cookies did they like? I mean.. Build a friendly working atmosphere with them for a comfortable co-operation, for a peer-to-peer.
2. Repeat the basics from yesterday. Create and remove some test files only using Unix-commands.
3. Open a test files using `vim` or `emacs`. Write some things. Google some hotkeys and commands for text editors which will increase your productivity.
4. Man new things for you. Type `say Hello world!`

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. Open YouTube. Type "learn c" in the search bar. Skip introduction and workspace setup parts and watch first 10-15 minutes of video.
2. Open `Terminal`. Create and open a new file `hello_world.c` using `vim` or `emacs`.
3. Write a "Hello world" program using `printf`. Save the file.
4. Compile this file as follows
   `clang -std=c11 -Wall -Wextra -Werror -Wpedantic hello_world.c -o hello_world`.
5. Run your program with `./hello_world`.
6. Try to write same program using `write` function instead of `printf`.
7. `the Auditor` is a style guide for `C`. You can find pdf of `the Auditor` in resources for this sprint. Keep in mind that you should follow `the Auditor` for all code you wrote in `C`.
8. You are ready to do a `t00`.

Task 00

NAME

Hello world

DIRECTORY

`t00/`

SUBMIT

`main.c`

ALLOWED FUNCTIONS

`printf`

DESCRIPTION

Write a program which will output a text below on the standard output followed by the newline.

CONSOLE OUTPUT

```
>clang -std=c11 -Wall -Wextra -Werror -Wpedantic -o hello_world main.c
>./hello_world | cat -e
Hello World$
>
```

FOLLOW THE WHITE RABBIT

`man 3 printf`

Task 01

NAME

Say wake up

DIRECTORY

`t01/`

SUBMIT

`mx_say_wake_up.c`

ALLOWED FUNCTIONS

`printf`

DESCRIPTION

Create a function which will output a text below on the standard output followed by the newline.

SYNOPSIS

```
void mx_say_wake_up(void);
```

CONSOLE OUTPUT

```
>./mx_say_wake_up | cat -e
Wake up, NEO \ (^_^) / ...$
The Matrix has you ...$
>
```

FOLLOW THE WHITE RABBIT

`man 3 printf`

Task 02

NAME

Write knock, knock

DIRECTORY

`t02/`

SUBMIT

`mx_write_knock_knock.c`

ALLOWED FUNCTIONS

write, strlen

DESCRIPTION

Create a function which will output a text below on the standard output followed by the newline.

SYNOPSIS

```
void mx_write_knock_knock(void);
```

CONSOLE OUTPUT

```
>./mx_write_knock_knock | cat -e
Follow the white rabbit.$
Knock, knock, Neo.$
>
```

FOLLOW THE WHITE RABBIT

man 2 write
man strlen

Task 03

NAME

Matrix voice

DIRECTORY

`t03/`

SUBMIT

`mx_matrix_voice.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output the smallest unit of matrix voice - beep (sound signal).

SYNOPSIS

```
void mx_matrix_voice(void);
```

CONSOLE OUTPUT

```
>./mx_matrix_voice | cat -e
^G%
>
```

FOLLOW THE WHITE RABBIT

```
man 2 write
man ascii
```

SEE ALSO

Matrix voice

Task 04

NAME

Print character

DIRECTORY

`t04/`

SUBMIT

`mx_printchar.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output single character on the standard output.

SYNOPSIS

```
void mx_printchar(char c);
```

FOLLOW THE WHITE RABBIT

```
man 2 write
man ascii
```

Task 05

NAME

Only printable

DIRECTORY

`t05/`

SUBMIT

`mx_only_printable.c, mx_printchar.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output all printable characters in reverse order on the standard output followed by the newline.

SYNOPSIS

```
void mx_only_printable(void);
```

FOLLOW THE WHITE RABBIT

`man ascii`

Task 06

NAME

Hexadecimal

DIRECTORY

`t06/`

SUBMIT

`mx_hexadecimal.c, mx_printchar.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output characters representing hexadecimal numerals in ascending order on the standard output followed by the newline. Characters should be in upper case.

SYNOPSIS

```
void mx_hexadecimal(void);
```

FOLLOW THE WHITE RABBIT

`man ascii`

Task 07

NAME

Print alphabet

DIRECTORY

`t07/`

SUBMIT

`mx_print_alphabet.c, mx_printchar.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output alphabet alternating upper and lower case characters in ascending order on the standard output followed by the newline. See example output below.

SYNOPSIS

```
void mx_print_alphabet(void);
```

EXAMPLE

```
mx_print_alphabet(); //prints AbC... ; there should be full alphabet instead of ...
```

FOLLOW THE WHITE RABBIT

`man ascii`

Task 08

NAME

String length

DIRECTORY

`t08/`

SUBMIT

`mx_strlen.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

```
int mx_strlen(const char *s);
```

FOLLOW THE WHITE RABBIT

man 3 strlen

Task 09

NAME

Print string

DIRECTORY

`t09/`

SUBMIT

`mx_printstr.c, mx_strlen.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will output string of characters on the standard output.

SYNOPSIS

```
void mx_printstr(const char *s);
```

FOLLOW THE WHITE RABBIT

`man 2 write`

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.