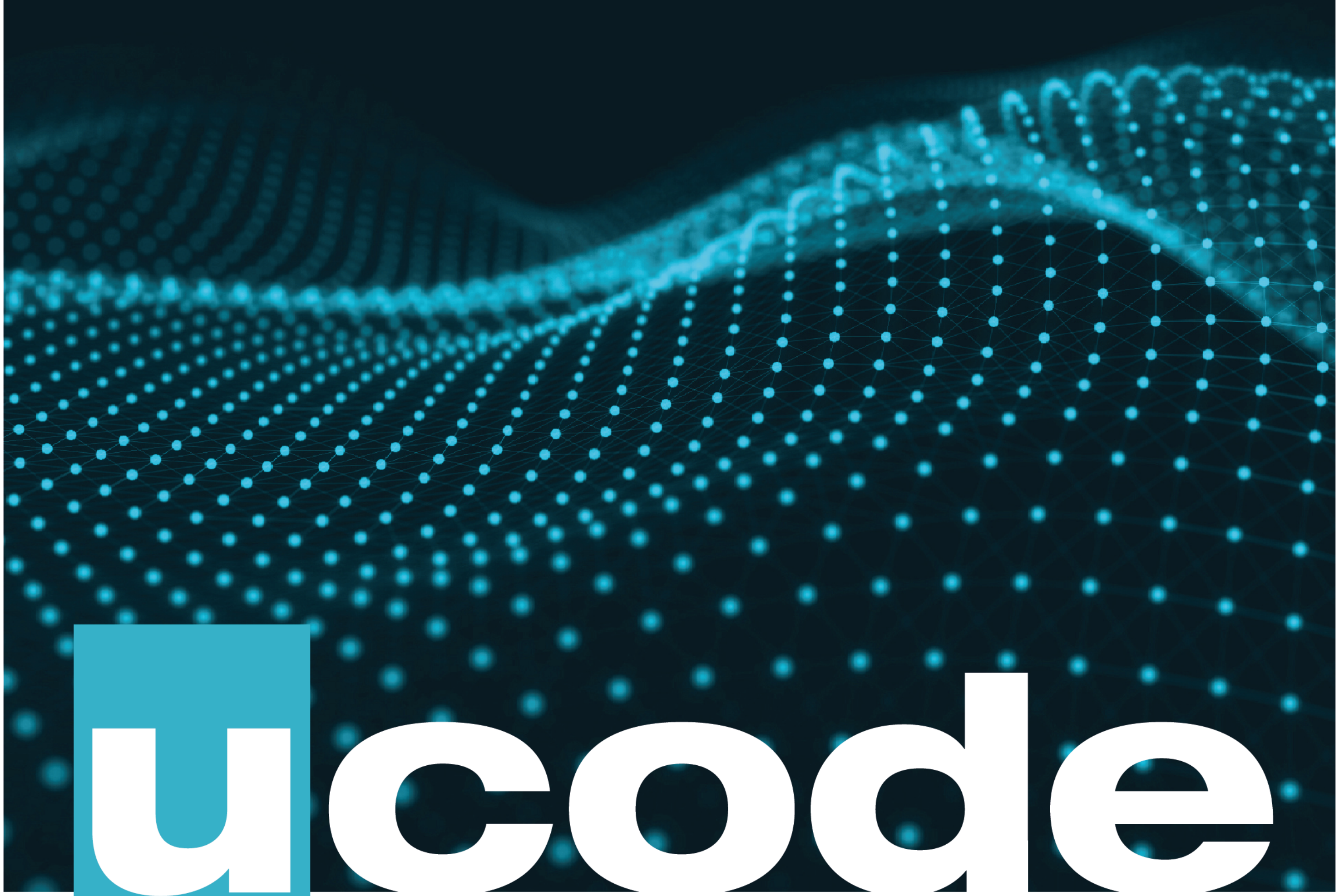
Sprint 04

Marathon C

October 15, 2019



Contents

ucode

Engage

DESCRIPTION

Hey up!

We hope that the previous days gave you a sense of confidence in the lines of code that you write.

Programming is primarily the optimization of various processes and actions. You always need to be ready to work with a lot of data and structuring information. For this, in `C`, there is such a data structure as arrays. They greatly simplify life and make the world a better place.

In this sprint, you will find a lot of information about arrays and derivative things from them.

BIG IDEA

Structuring data in the program.

ESSENTIAL QUESTION

What are the ways to competently manage data in `C`?

CHALLENGE

Learn to use arrays.

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.

- Well, have you liked the task with the pointers?
- How was your sprint yesterday? How much tasks have you done?
- What topics were unclear to you?
- What is array?
- What is a dimension in arrays understanding?
- What is segmentation fault?
- What is sorting? What are the simplest algorithms for sorting numbers?
- How to use algorithms efficiently?
- What difference between a character array and a string?

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. Repeat everything you know and do not know about the pointers, today they also need for you.
2. Find all information about arrays in `C`. Use arrays in practice.
3. Create a few-dimensional array. Do you know how to fill an array with more than one dimension?
4. Just have fun with all this stuff ;)

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 evaluated 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.
- Google every new word you have not heard before.
- 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.
2. Only what you pushed into your repository will be evaluated.
3. Communicate and brainstorm with other students.
4. You have 40 hours to overcome this challenge.

Task 00

NAME

Print array

DIRECTORY

`t00/`

SUBMIT

`mx_print_arr_int.c, mx_printint.c, mx_printchar.c`

ALLOWED FUNCTIONS

`write`

DESCRIPTION

Create a function which will print all array numbers on the standard output.
Each number must be followed by the newline.

SYNOPSIS

```
void mx_print_arr_int(const int *arr, int size);
```

Task 01

NAME

Square root

DIRECTORY

`t01/`

SUBMIT

`mx_sqrt.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will compute the non-negative square root of `x`.
Function must compute square root in less than 2 seconds.

RETURN

It returns the square root of the number `x` if it's natural, and `0` otherwise.

SYNOPSIS

```
int mx_sqrt(int x);
```

EXAMPLE

```
mx_sqrt(3); //returns 0
mx_sqrt(4); //returns 2
```

FOLLOW THE WHITE RABBIT

man time

Task 02

NAME

Locate character

DIRECTORY

`t02/`

SUBMIT

`mx_strchr.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

```
char *mx_strchr(const char *s, int c);
```

FOLLOW THE WHITE RABBIT

man strchr

Task 03

NAME

Copy them all

DIRECTORY

`t03/`

SUBMIT

`mx_strncpy.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

RETURN

Returns `pointer` to the first element of `dst`.

SYNOPSIS

```
char *mx_strncpy(char *dst, const char *src, int len);
```

EXAMPLE

```
src[11] = "yo neo bro";
dst[11];
mx_strncpy(dst, src, 3); //dst now is "yo "
```

Task 04

NAME

Concatenate strings

DIRECTORY

`t04/`

SUBMIT

`mx_strcat.c, mx_strlen.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

man strcat

Task 05

NAME

Sort array

DIRECTORY

`t05/`

SUBMIT

`mx_sort_arr_int.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will sort array of integers in ascending order.

SYNOPSIS

```
void mx_sort_arr_int(int *arr, int size);
```

EXAMPLE

```
arr = {3, 55, -11, 1, 0, 4, 22};
mx_sort_arr_int(arr, 7); //arr now is '{-11, 0, 1, 3, 4, 22, 55}'
```

Task 06

NAME

ASCII to integer

DIRECTORY

`t06/`

SUBMIT

`mx_atoi.c, mx_isdigit.c, mx_isspace.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will convert ASCII string to integer.

SYNOPSIS

```
int mx_atoi(const char *str);
```

FOLLOW THE WHITE RABBIT

man atoi

Task 07

NAME

Count words

DIRECTORY

`t07/`

SUBMIT

`mx_count_words.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will count words in the string.
Word is a sequence of characters separated by delimiter.

RETURN

Returns number of words in the string.

SYNOPSIS

```
int mx_count_words(const char *str, char delimiter);
```

EXAMPLE

```
str = "  follow  *   the  white rabbit ";
mx_count_words(str, '*'); //returns 2
mx_count_words(str, ' '); //returns 5
```

Task 08

NAME

Popular number

DIRECTORY

`t08/`

SUBMIT

`mx_popular_int.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will find the most common number in array of integers.

RETURN

- return the most common number in array of integers;
- if there are more than one the most common numbers return the first number met in array.

SYNOPSIS

```
int mx_popular_int(const int *arr, int size);
```

EXAMPLE

```
arr = {2, 2, 4, 4};
mx_popular_int(arr, 4); //returns 2
```

Task 09

NAME

Compare strings N

DIRECTORY

`t09/`

SUBMIT

`mx_strncmp.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

```
int mx_strncmp(const char *s1, const char *s2, int n);
```

FOLLOW THE WHITE RABBIT

`man strncmp`

Task 10

NAME

Locate a substring

DIRECTORY

`t10/`

SUBMIT

`mx_strstr.c, mx_strlen.c, mx_strncmp.c, mx_strchr.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

`man strstr`

Task 11

NAME

Count substrings

DIRECTORY

`t11/`

SUBMIT

`mx_count_substr.c, mx_strstr.c, mx_strlen.c, mx_strncmp.c, mx_strchr.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function which will count substrings `sub` in string `str`.

RETURN

- return count of `sub` in `str`;
- in case `sub` is the empty string function returns `0`.

SYNOPSIS

```
int mx_count_substr(const char *str, const char *sub);
```

EXAMPLE

```
str = "yo, yo, yo Neo";
sub = "yo";
mx_count_substr(str, sub); //returns 3
```

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.