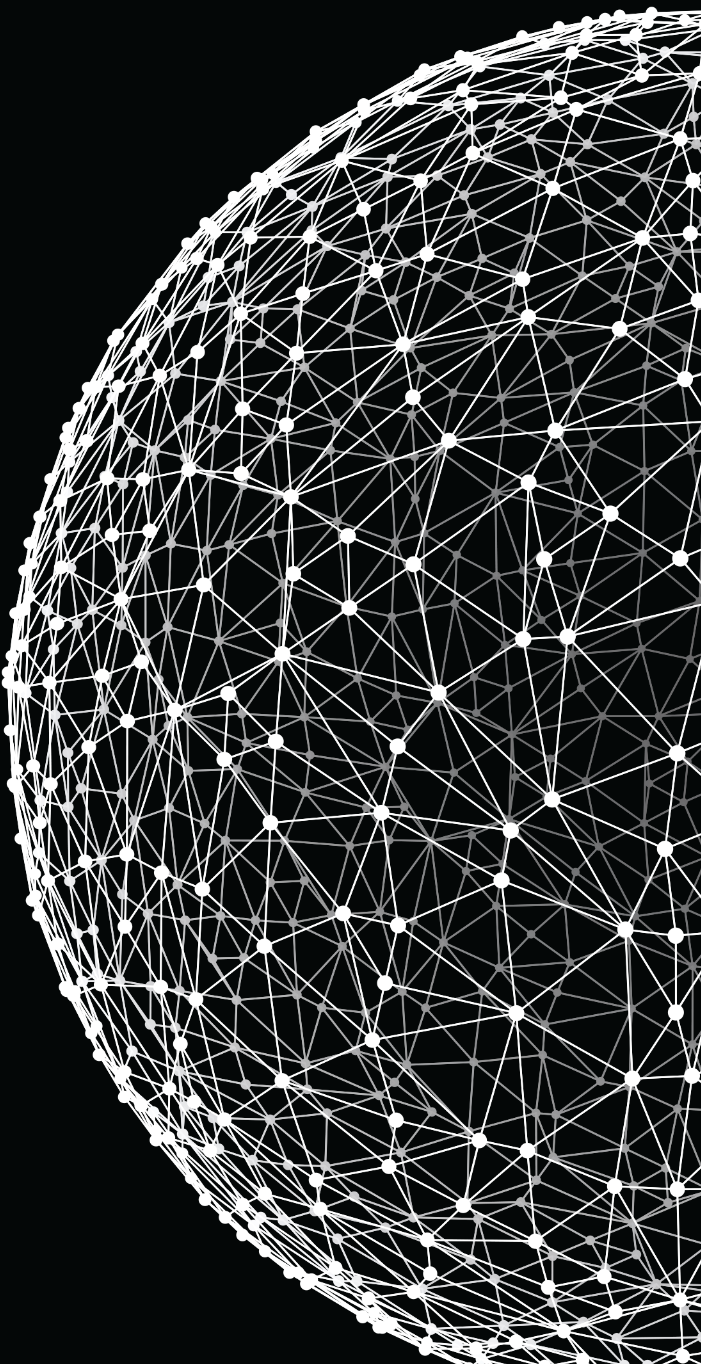
Sprint 04

Marathon C

October 13, 2020

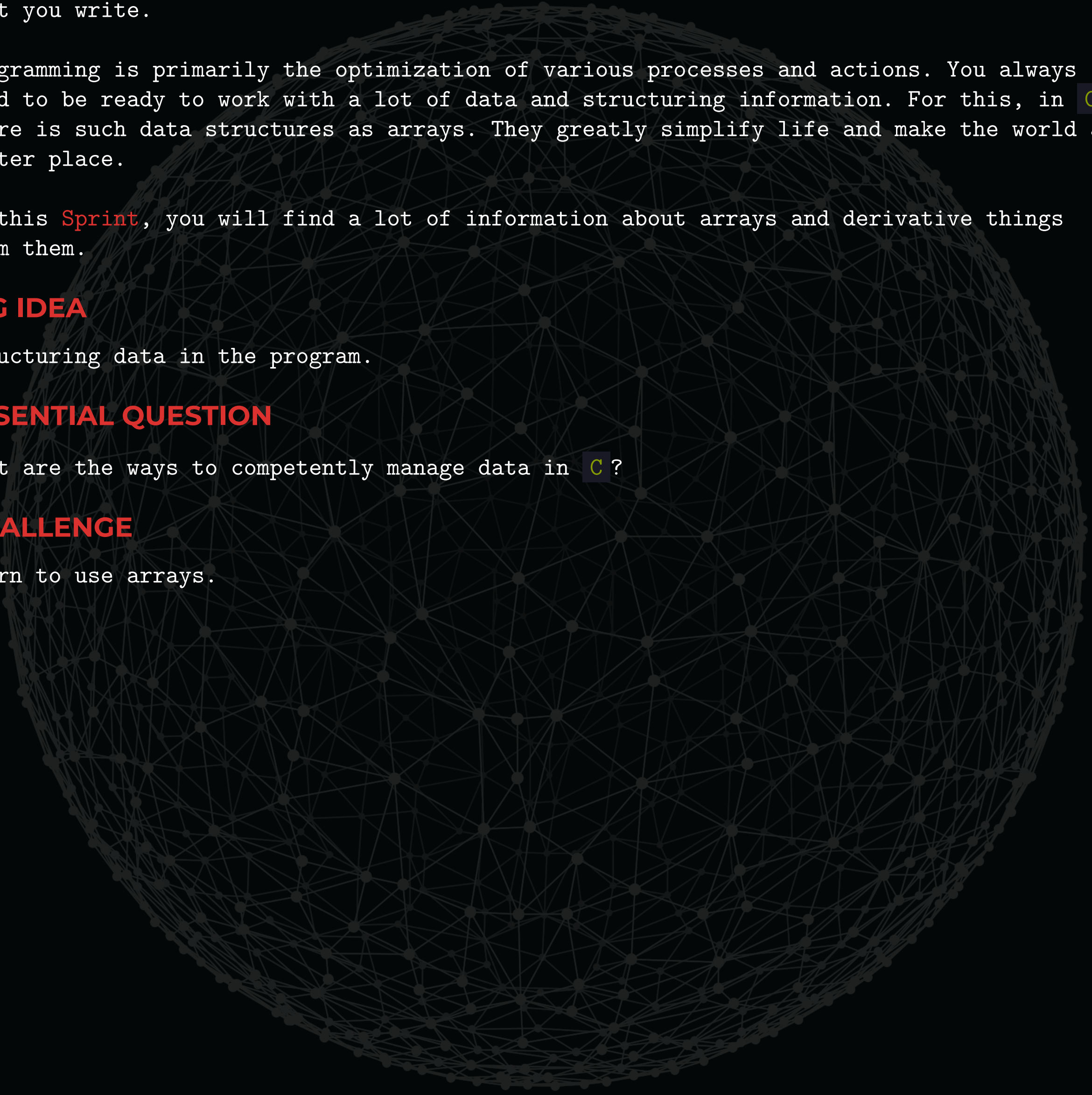


ucode connect

Contents

ucode connect

Engage



DESCRIPTION

Greetings!

We hope that the previous challenges 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 data structures 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.

ucode connect

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- Did you enjoy the tasks on pointers?
- How was your previous Sprint? How many tasks have you completed?
- What topics were unclear to you?
- What is an array?
- What is a dimension in arrays understanding?
- What is a segmentation fault?
- What is sorting? What are the simplest algorithms for sorting numbers?
- How to use algorithms efficiently?
- What is the difference between a character array and a string?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

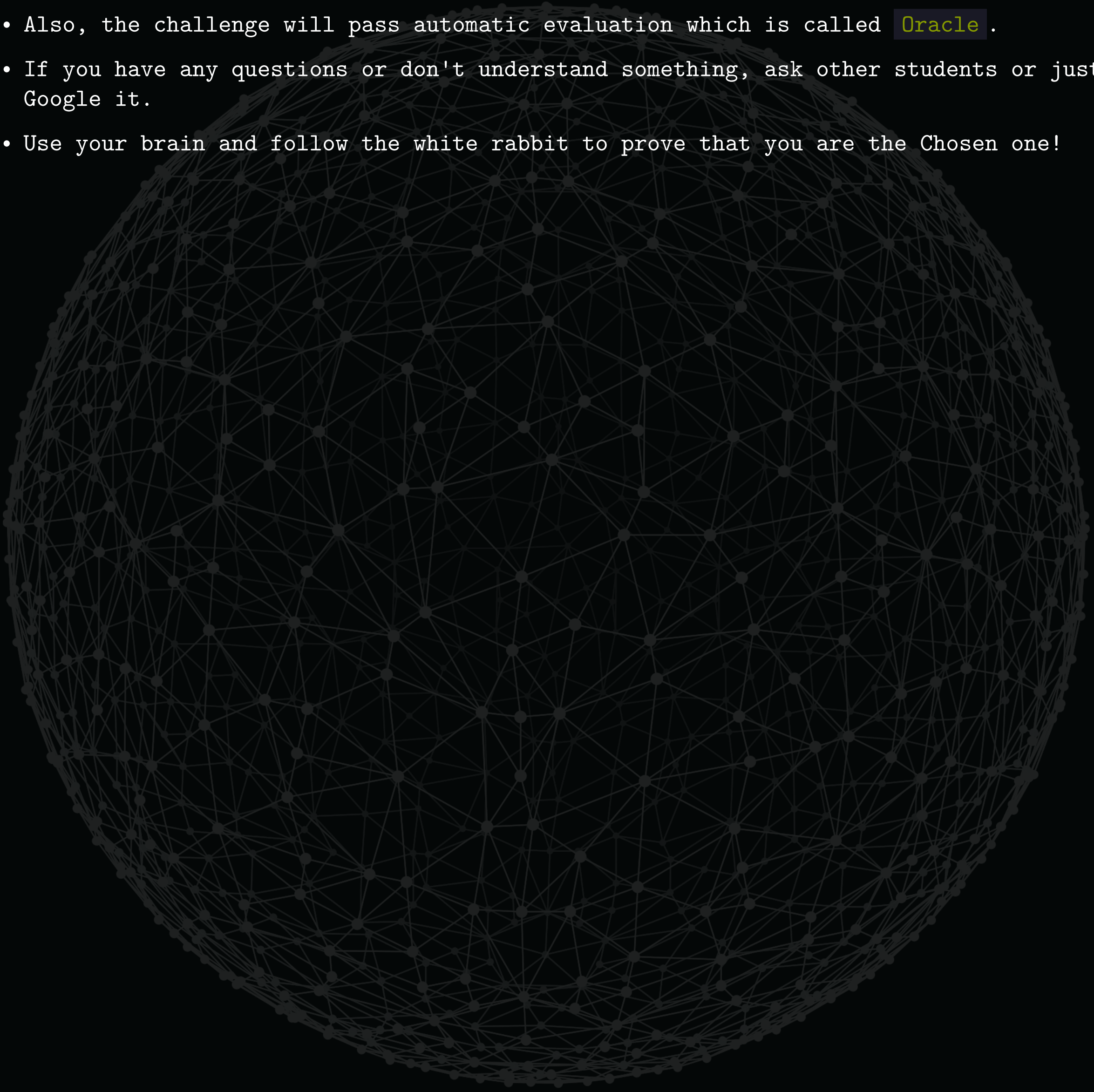
- Repeat the basics from the previous challenges. Repeat everything you know and do not know about pointers, because you will need them in this challenge as well.
- Find information about arrays in `C`. Use arrays in practice.
- Create a multi-dimensional array. Do you know how to fill an array with more than one dimension?
- Clone your git repository that is issued on the challenge page in the LMS. Use `git clone` for this.
- Proceed to the tasks.
- Arrange to brainstorm tasks with other students.
- Try to implement your thoughts in code.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- 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 this document.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Compile C-files with clang compiler and use these flags: `clang -std=c11 -Wall -Wextra -Werror -Wpedantic`.

- Pay attention to what is allowed in a certain task. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- Complete tasks according to the rules specified in the `Auditor` .
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Also, the challenge will pass automatic evaluation which is called `Oracle` .
- If you have any questions or don't understand something, ask other students or just Google it.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!



Act Basic: 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 that prints all array numbers to the standard output.
Each number must be followed by a newline.

SYNOPSIS

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

ucode connect

Act Basic: Task 01

NAME

Square root

DIRECTORY

`t01/`

SUBMIT

`mx_sqrt.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that computes the non-negative square root of `x` .
The function must compute square root in less than 2 seconds.

RETURN

Returns the square root of the number `x` if it is 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

Act Basic: Task 02

NAME

Locate character

DIRECTORY

`t02/`

SUBMIT

`mx_strchr.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

`man strchr`

Act Basic: Task 03

NAME

Copy them all

DIRECTORY

`t03/`

SUBMIT

`mx_strncpy.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

RETURN

Returns a `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 "
```

Act Basic: Task 04

NAME

Concatenate strings

DIRECTORY

`t04/`

SUBMIT

`mx_strcat.c, mx_strlen.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

`man strcat`

Act Basic: Task 05

NAME

Sort array

DIRECTORY

`t05/`

SUBMIT

`mx_sort_arr_int.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that sorts an 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}'
```

Act Basic: 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 that converts an ASCII string to an integer as the standard libc function `atoi` does.

Take into account function's work with overflows.

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

man atoi

Act Basic: Task 07

NAME

Count words

DIRECTORY

`t07/`

SUBMIT

`mx_count_words.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that counts words in a string.
Word is a sequence of characters separated by a delimiter.

RETURN

Returns the number of words in the string.

SYNOPSIS

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

EXAMPLE

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

Act Basic: Task 08

NAME

Popular number

DIRECTORY

`t08/`

SUBMIT

`mx_popular_int.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

Create a function that finds the most common number in an array of integers.

RETURN

- Returns the most common number in the array of integers
- Returns the first number found in the array if there is more than one most common number

SYNOPSIS

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

EXAMPLE

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

Act Basic: Task 09

NAME

Compare strings N

DIRECTORY

`t09/`

SUBMIT

`mx_strncmp.c`

ALLOWED FUNCTIONS

None

DESCRIPTION

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

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

`man strncmp`

Act Basic: 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 that has the same behaviour as the standard libc function `strstr` .

SYNOPSIS

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

FOLLOW THE WHITE RABBIT

man strstr

Act Basic: 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 that counts the substrings `sub` in the string `str` .

RETURN

- Returns the count of `sub` in `str`
- Returns `0` if `sub` is an empty string

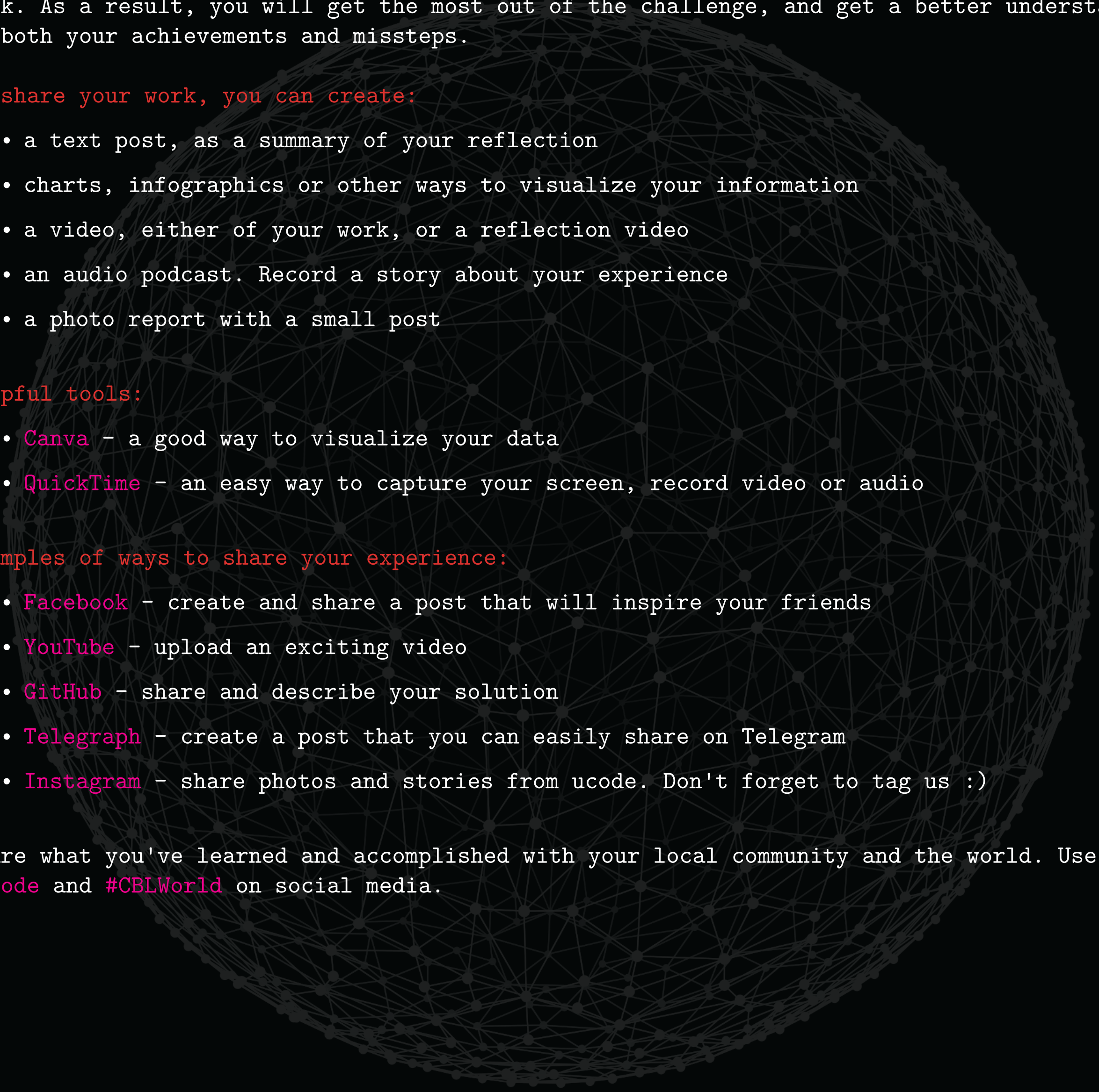
SYNOPSIS

```
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

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

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

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

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