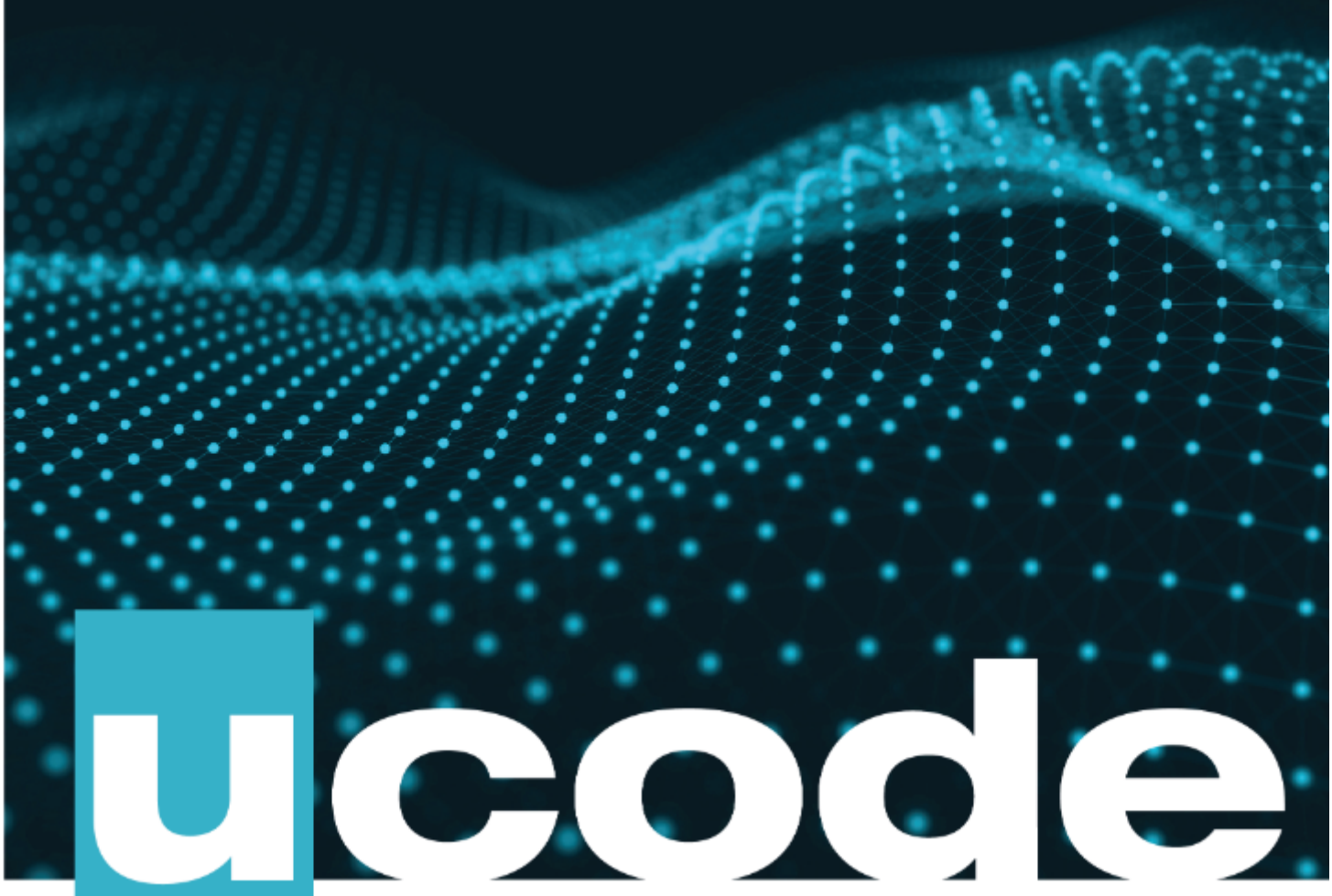
# Sprint 02

Marathon C

April 15, 2020



ucode

# Contents



ucode

# Engage

## DESCRIPTION

Ah C, here we go again.

We hope that your first acquaintance with the C language was successful.

Using standard functions makes writing code easier, but we believe that it is better to discover how they work by writing the algorithms behind common simple function by yourself.

Imagine how great it is to understand why a particular function works in a certain way.

In this Sprint, you will rewrite some standard functions and implement the basics of mathematics in C.

## BIG IDEA

Learn to use C to solve real-world problems.

## ESSENTIAL QUESTION

How to maximize the benefits of Peer-to-Peer?

## CHALLENGE

Delve into the C language.

# 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 and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- Do you like C?
- How was the Sprint yesterday? How many tasks have you completed?
- What topics were unclear to you?
- What is a function? What is a loop?
- How to use standard functions?
- Do you know what standard output is?
- How to print text to the console?
- How many letters are there in the English alphabet?
- What is the difference between a digit and a number?

## 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 yesterday. Try to output your name to the standard output using C. And also using the Unix command.
- Try to compile the written code and run your program. Does it work as you expected?
- Use some standard functions. Just for fun. Try to use `getchar`, `gets` and `puts`.
- Clone your git repository that is issued on the challenge page in the LMS. Use `git clone` for this.
- Open the story and read it!
- 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: Task 00

## NAME

Positive or negative

## DIRECTORY

`t00/`

## SUBMIT

`mx_is_positive.c, mx_printstr.c, mx_strlen.c`

## ALLOWED FUNCTIONS

`write`

## DESCRIPTION

Create a function that outputs `positive`, `negative` or `zero` followed by a newline to standard output whether number is positive, negative or equal to 0.

## SYNOPSIS

```
void mx_is_positive(int i);
```

## EXAMPLE

```
mx_is_positive(2); //prints positive
```

## FOLLOW THE WHITE RABBIT

`man 2 write`

# Act: Task 01

## NAME

Odd or even

## DIRECTORY

`t01/`

## SUBMIT

`mx_is_odd.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function that checks whether a number is odd or even.

## RETURN

Returns `true` if number is odd or `false` if number is even.

## SYNOPSIS

```
bool mx_is_odd(int value);
```

## EXAMPLE

```
mx_is_odd(1);  //returns true
```

# Act: Task 02

## NAME

Is alphabetic?

## DIRECTORY

`t02/`

## SUBMIT

`mx_isalpha.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
bool mx_isalpha(int c);
```

## EXAMPLE

```
mx_isalpha('a');  //returns 1
```

## FOLLOW THE WHITE RABBIT

man isalpha

# Act: Task 03

## NAME

Is digit?

## DIRECTORY

`t03/`

## SUBMIT

`mx_isdigit.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
bool mx_isdigit(int c);
```

## EXAMPLE

```
mx_isdigit('A');  //returns 0
```

## FOLLOW THE WHITE RABBIT

```
man isdigit
```

## NAME

## DIRECTORY

`t04/`

## SUBMIT

`mx_isspace.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

`isspace`

## SYNOPSIS

```
bool mx_isspace(char c);
```

## EXAMPLE

```
mx_isspace(' ');  //returns 1
```

## FOLLOW THE WHITE RABBIT

man isspace

## SEE ALSO

Whitespace character

# Act: Task 05

## NAME

Is lower case?

## DIRECTORY

`t05/`

## SUBMIT

`mx_islower.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
bool mx_islower(int c);
```

## EXAMPLE

```
mx_islower('Z');  //returns 0
```

## FOLLOW THE WHITE RABBIT

man islower

# Act: Task 06

## NAME

Is upper case?

## DIRECTORY

`t06/`

## SUBMIT

`mx_isupper.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
bool mx_isupper(int c);
```

## EXAMPLE

```
mx_isupper('Z');  //returns 1
```

## FOLLOW THE WHITE RABBIT

```
man isupper
```

# Act: Task 07

## NAME

To lower case

## DIRECTORY

`t07/`

## SUBMIT

`mx_tolower.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
int mx_tolower(int c);
```

## EXAMPLE

```
mx_tolower('Z');  //returns z
mx_tolower('z');  //returns z
```

## FOLLOW THE WHITE RABBIT

man tolower

# Act: Task 08

## NAME

To upper case

## DIRECTORY

`t08/`

## SUBMIT

`mx_toupper.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

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

## SYNOPSIS

```
int mx_toupper(int c);
```

## EXAMPLE

```
mx_toupper('Z');  //returns Z
mx_toupper('z');  //returns Z
```

## FOLLOW THE WHITE RABBIT

```
man toupper
```

# Act: Task 09

## NAME

Isosceles triangle

## DIRECTORY

`t09/`

## SUBMIT

`mx_isos_triangle.c, mx_printchar.c`

## ALLOWED FUNCTIONS

`write`

## DESCRIPTION

Create a function that outputs to standard output an isosceles triangle:

- with a given triangle side length and a character to fill the figure
- each row must be followed by a newline

## SYNOPSIS

```
void mx_isos_triangle(unsigned int length, char c);
```

## CONSOLE OUTPUT

```
>./mx_isos_triangle | cat -e    # for mx_isos_triangle(3, '*');
*$
**$
***$
>
```

# Act: Task 10

## NAME

Multiple of a number

## DIRECTORY

`t10/`

## SUBMIT

`mx_multiple_number.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function that checks whether a natural number `mult` is a multiple of a number `n`.

## RETURN

Returns `true` if the number `mult` is the multiple of the number `n`, otherwise `false`.

## SYNOPSIS

```
bool mx_multiple_number(int n, int mult);
```

## EXAMPLE

```
mx_multiple_number(3, 9); //returns true
```

# Act: Task 11

## NAME

Find maximum

## DIRECTORY

`t11/`

## SUBMIT

`mx_max.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function that finds the largest number among options.

## RETURN

Returns value of the maximum number.

## SYNOPSIS

```
int mx_max(int a, int b, int c);
```

## EXAMPLE

```
mx_max(-1, 0, 1); //returns 1
```

# Act: Task 12

## NAME

Middle number

## DIRECTORY

`t12/`

## SUBMIT

`mx_mid.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function that finds the middle number among options.

## RETURN

Returns value of the middle number.

## SYNOPSIS

```
int mx_mid(int a, int b, int c);
```

## EXAMPLE

```
mx_mid(5, 16, 10); //returns 10
mx_mid(5, 6, 6); //returns 6
```

# Act: Task 13

## NAME

Sum digits

## DIRECTORY

`t13/`

## SUBMIT

`mx_sum_digits.c`

## ALLOWED FUNCTIONS

None

## DESCRIPTION

Create a function that sums up all digits of a number.

## RETURN

Returns the sum of all digits of the number.

## SYNOPSIS

```c
int mx_sum_digits(int num);
```

## EXAMPLE

```c
mx_sum_digits(435); //returns 12
mx_sum_digits(-555); //returns 15
```

# Act: Task 14

## NAME

Print integer

## DIRECTORY

`t14/`

## SUBMIT

`mx_printint.c, mx_printchar.c`

## ALLOWED FUNCTIONS

`write`

## DESCRIPTION

Create a function that outputs integer values to standard output.

## SYNOPSIS

```
void mx_printint(int n);
```

## EXAMPLE

```
mx_printint(25); //prints 25
mx_printint(2147483647); //prints 2147483647
```