

Version: 1.0



# Selection

## C-Simple-String

### Summary

This project focuses on implementing basic string manipulation functions in C, covering alphabetic checks, numeric checks, case transformations, and string length calculations.

#C

#String

#Functions

# 42

# Intellectual Property Disclaimer

All content presented in this training module, including but not limited to texts, images, graphics, and other materials, is protected by intellectual property rights held by Association 42.

## Terms of Use:

- **Personal use:** You are permitted to use the contents of this module solely for personal purpose. Any commercial use, reproduction, distribution, modification, or public display is strictly prohibited without prior written permission from Association 42.
- **Respect for Integrity:** You must not alter, transform, or adapt the content in any way that could harm its integrity.

## Protection of Rights:

Any violation of these terms constitutes an infringement of intellectual property rights and may result in legal action. We reserve the right to take all necessary measures to protect our rights, including but not limited to claims for damages.

*For any questions regarding the use of the content or to obtain authorization, please contact:*  
legal@42.fr

# Contents

<b>1</b>	<b>Instructions</b>	<b>1</b>
<b>2</b>	<b>Foreword</b>	<b>5</b>
<b>3</b>	<b>Exercise 0: ft_str_is_alpha</b>	<b>6</b>
<b>4</b>	<b>Exercise 1: ft_str_is_numeric</b>	<b>7</b>
<b>5</b>	<b>Exercise 2: ft_str_is_lowercase</b>	<b>8</b>
<b>6</b>	<b>Exercise 3: ft_str_is_uppercase</b>	<b>9</b>
<b>7</b>	<b>Exercise 4: ft_str_is_printable</b>	<b>10</b>
<b>8</b>	<b>Exercise 5: ft_strupcase</b>	<b>11</b>
<b>9</b>	<b>Exercise 6: ft_strlowercase</b>	<b>12</b>
<b>10</b>	<b>Exercise 7: ft_strcapitalize</b>	<b>13</b>
<b>11</b>	<b>Exercise 8: ft_strlen</b>	<b>14</b>
<b>12</b>	<b>Exercise 9: ft_putstr</b>	<b>15</b>
<b>13</b>	<b>Exercise 10: ft_putstr_non_printable</b>	<b>16</b>
<b>14</b>	<b>Submission and peer-evaluation</b>	<b>17</b>

# Chapter 1

## Instructions

- Only this page will serve as a reference: do not trust rumors.
- Watch out! This document could potentially change up until submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- Additionally, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated, and there is no way to negotiate with it. So, to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try to understand your code if it doesn't adhere to the Norm. Moulinette relies on a program called `norminette` to check if your files respect the norm. TL;DR: it would be foolish to submit work that doesn't pass `norminette`'s check.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We **will not** consider a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- You'll only have to submit a `main()` function if we ask for a program.
- Moulinette compiles with these flags: `-Wall -Wextra -Werror`, and uses `cc`.
- If your program doesn't compile, you'll get 0.
- You cannot leave any additional files in your directory other than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ...`
- Check out the Slack Piscine.

- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor! Use your brain!!!



Do not forget to add the *standard 42 header* in each of your `.c/.h` files. Norminette checks its existence anyway!



Norminette must be launched with the `-R CheckForbiddenSourceHeader` flag. Moulinette will use it too.

## ● Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

## ● Main message

- 👉 Build strong foundations without shortcuts.
- 👉 Really develop tech & power skills.
- 👉 Experience real peer-learning, start learning how to learn and solve new problems.
- 👉 The learning journey is more important than the result.
- 👉 Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

## ● Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

## ● Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

## ● Comments and example:

- Yes, we know AI exists — and yes, it can solve your activities. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

### ✓ Good practice:

I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

### ✗ Bad practice:

I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

## Chapter 2

### Foreword

#### A Fun Fact About the Number 42

The number 42 holds a special place in both mathematics and popular culture. Here's a fascinating mathematical fact about 42:

In base 10, 42 is an even integer, but it also has interesting properties in other bases. For example, in base 13, the number 42 is written as  $33_{13}$ , which is a repunit (a number consisting of repeated units).

Additionally, 42 is a pronic number, which means it is the product of two consecutive integers:  $6 \times 7 = 42$ . Pronic numbers are also known as oblong numbers or rectangular numbers.

In the realm of geometry, a polygon with 42 sides is called a tetracontakaidigon. While not commonly discussed, it's intriguing to consider the properties of such a shape.

Moreover, 42 is a sphenic number, which means it is the product of three distinct prime numbers:  $2 \times 3 \times 7 = 42$ . Sphenic numbers have unique factorization properties that make them interesting to study in number theory.


Beyond mathematics, the number 42 is famously known as the "Answer to the Ultimate Question of Life, The Universe, and Everything" in Douglas Adams' science fiction series "The Hitchhiker's Guide to the Galaxy." This reference has cemented 42 as a beloved number in popular culture, often appearing in various contexts as an inside joke or homage.

In summary, whether you're exploring its mathematical properties or enjoying its cultural significance, the number 42 offers a wealth of intriguing insights and fun trivia.



## Chapter 3

### Exercise 0: ft\_str\_is\_alpha

	Exercise: 0	
ft_str_is_alpha		
Directory: ex0/		
Files to Submit: ft_str_is_alpha.c		
Authorized: None		

- Create a function that returns 1 if the string given as a parameter contains only alphabetical characters, and 0 if it contains any other character.


**Prototype:**

```
int ft_str_is_alpha(char *str);
```

- It should return 1 if str is empty.

## Chapter 4

### Exercise 1: ft\_str\_is\_numeric

	Exercise: 1	
ft_str_is_numeric		
Directory: ex1/		
Files to Submit: ft_str_is_numeric.c		
Authorized: None		

- Create a function that returns 1 if the string given as a parameter contains only digits, and 0 if it contains any other character.


**Prototype:**

```
int ft_str_is_numeric(char *str);
```

- It should return 1 if str is empty.

## Chapter 5

### Exercise 2: ft\_str\_is\_lowercase

	Exercise: 2	
ft_str_is_lowercase		
Directory: ex2/		
Files to Submit: ft_str_is_lowercase.c		
Authorized: None		

- Create a function that returns 1 if the string given as a parameter contains only lowercase alphabetical characters, and 0 if it contains any other character.


#### Prototype:

```
int ft_str_is_lowercase(char *str);
```

- It should return 1 if str is empty.

## Chapter 6

### Exercise 3: ft\_str\_is\_uppercase

	Exercise: 3	
ft_str_is_uppercase		
Directory: ex3/		
Files to Submit: ft_str_is_uppercase.c		
Authorized: None		

- Create a function that returns 1 if the string given as a parameter contains only uppercase alphabetical characters, and 0 if it contains any other character.


**Prototype:**

```
int ft_str_is_uppercase(char *str);
```

- It should return 1 if str is empty.

## Chapter 7

### Exercise 4: ft\_str\_is\_printable

	Exercise: 4	
ft_str_is_printable		
Directory: ex4/		
Files to Submit: ft_str_is_printable.c		
Authorized: None		

- Create a function that returns 1 if the string given as a parameter contains only printable characters, and 0 if it contains any other character.


#### Prototype:

```
int ft_str_is_printable(char *str);
```

- It should return 1 if str is empty.

## Chapter 8

### Exercise 5: ft\_strupcase

	Exercise: 5	
ft_strupcase		
Directory: ex5/		
Files to Submit: ft_strupcase.c		
Authorized: None		

- Create a function that transforms every letter to uppercase.


**Prototype:**

```
char *ft_strupcase(char *str);
```

- It should return str.

## Chapter 9

### Exercise 6: ft\_strlowercase

	Exercise: 6	
ft_strlowercase		
Directory: ex6/		
Files to Submit: ft_strlowercase.c		
Authorized: None		

- Create a function that transforms every letter to lowercase.


**Prototype:**

```
char *ft_strlowercase(char *str);
```

- It should return str.

## Chapter 10

### Exercise 7: ft\_strcapitalize

	Exercise: 7	
ft_strcapitalize		
Directory: ex7/		
Files to Submit: ft_strcapitalize.c		
Authorized: None		

- Create a function that capitalizes the first letter of each word and transforms all other letters to lowercase.
- A word is a string of alphanumeric characters.

#### Prototype:

```
char *ft_strcapitalize(char *str);
```

- It should return str.
- For example:


#### Example:

```
hello, how are you doing? 42words forty-two; fifty+and+one
Hello, How Are You Doing? 42words Forty-Two; Fifty+And+One
```



## Chapter 11

### Exercise 8: ft\_strlen

	Exercise: 8	
ft_strlen		
Directory: ex8/		
Files to Submit: ft_strlen.c		
Authorized: None		


- Create a function that counts and returns the number of characters in a string.

**Prototype:**

```
int ft_strlen(char *str);
```

## Chapter 12

### Exercise 9: ft\_putstr

	Exercise: 9	
ft_putstr		
Directory: ex9/		
Files to Submit: ft_putstr.c		
Authorized: write		


- Create a function that displays a string of characters on the standard output.

**Prototype:**

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

## Chapter 13

### Exercise 10: ft\_putstr\_non\_printable

	Exercise: 10	
ft_putstr_non_printable		
Directory: ex10/		
Files to Submit: ft_putstr_non_printable.c		
Authorized: write		

- Create a function that displays a string of characters onscreen. If this string contains characters that aren't printable, they'll have to be displayed in the shape of hexadecimals (lowercase), preceded by a "backslash".
- For example:

```
Hello\nHow are you ?
```

- The function should display:

```
Hello\0aHow are you ?
```

#### Prototype:

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

## Chapter 14

# Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.



You need to return only the files requested by the subject of this project.