Version: 1.1

Printf

Because ft_putnbr() and ft_putstr() aren't enough

Summary

The goal of this project is pretty straightforward. You will recode printf(). You will mainly learn about using a variable number of arguments. How cool is that? It is actually pretty cool :)

#C

#Imperative

#Library

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

Chapter 1

Introduction

You will discover a popular and versatile C function: `printf()`. This exercise is a great opportunity to improve your programming skills. It is of moderate difficulty.

You will discover **variadic functions** in C.

The key to a successful `ft_printf` is a well-structured and extensible code.

Once this assignment is passed, you will be allowed to add your `ft_printf()` to your `libft` so you can use it in your C projects at 42.

Chapter 2

Common Instructions

- Your activity must be written in C.
- Your activity must be written in accordance with the Norm. If you have bonus files or functions, they are included in the norm check, and you will receive a 0 if there is a norm error inside.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc.) except for undefined behaviors. If this happens, your activity will be considered non-functional and will receive a 0 during the review.
- All heap-allocated memory space must be properly freed when necessary. No memory leaks will be tolerated.
- If the subject requires it, you must submit a `Makefile` that will compile your source files to the required output with the flags `-Wall`, `-Wextra`, and `-Werror`, using `cc`, and your `Makefile` must not relink.
- Your `Makefile` must at least contain the rules `$(NAME)`, `all`, `clean`, `fclean`, and `re`.
- If your activity allows you to use your `libft`, you must copy its sources and its associated `Makefile` into a `libft` folder. Your activity's `Makefile` must compile the library by using its `Makefile`, then compile the activity.
- We encourage you to create test programs for your activity, even though this work **will not be submitted and will not be graded**. It will give you a chance to easily test your work and your peers' work. You will find these tests especially useful during your defense. Indeed, during the defense, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned Git repository. Only the work in the Git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer evaluations. If an error occurs in any section of your work during Deepthought's grading, the review will stop.

© 2025 Association 42. - All rights reserved

Chapter 3

AI Instructions

Context

This activity is designed to help you discover the fundamental building blocks of your 42 training.

To properly anchor key knowledge and skills, it's essential to adopt a thoughtful approach to using AI tools and support.

True foundational learning requires genuine intellectual effort — through challenge, repetition, and peer-learning exchanges.

For a more complete overview of our stance on AI — as a learning tool, as part of the 42 training, and as an expectation in the job market — please refer to the dedicated FAQ 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.

© 2025 Association 42. - All rights reserved

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.

© 2025 Association 42. - All rights reserved

Chapter 4

Mandatory part

Program Name	`libftprintf.a`
Files to Submit	`Makefile, *.h, */*.h, *.c, */*.c`
Makefile	`NAME, all, clean, fclean, re`
External Functions	`malloc, free, write,` `va_start, va_arg, va_copy, va_end`
Libft authorized	`Yes`
Description	`Write a library that contains ft_printf(), a function that will mimic the original printf()`

You have to recode the `printf()` function from `libc`.

The prototype of `ft_printf()` is:

```
int    ft_printf(const char *, ...);
```

Here are the requirements:

- Don't implement the buffer management of the original `printf()`.
- Your function has to handle the following conversions: `cspdiuxX%`
- Your function will be compared against the original `printf()`.
- You must use the command `ar` to create your library.
  Using the `libtool` command is forbidden.
- Your `libftprintf.a` has to be created at the root of your repository.
- Your header file must be named `ft_printf.h` and must contain the prototype of your `ft_printf()` function.

You have to implement the following conversions:

- `%c` Prints a single character.
- `%s` Prints a string (as defined by the common `C` convention).
- `%p` The `void *` pointer argument has to be printed in hexadecimal format.
- `%d` Prints a decimal (base 10) number.
- `%i` Prints an integer in base 10.
- `%u` Prints an unsigned decimal (base 10) number.
- `%x` Prints a number in hexadecimal (base 16) lowercase format.
- `%X` Prints a number in hexadecimal (base 16) uppercase format.
- `%%` Prints a percent sign.

© 2025 Association 42. - All rights reserved

Chapter 5

Readme Requirements

A README.md file must be provided at the root of your Git repository. Its purpose is to allow anyone unfamiliar with the activity (peers, staff, recruiters, etc.) to quickly understand what the activity is about, how to run it, and where to find more information on the topic.
The README.md must include at least:

- The very first line must be italicized and read: *This activity has been created as part of the 42 curriculum by <login1>[, <login2>[, <login3>[...]]].*
- A "**Description**" section that clearly presents the activity, including its goal and a brief overview.
- An "**Instructions**" section containing any relevant information about compilation, installation, and/or execution.
- A "**Resources**" section listing classic references related to the topic (documentation, articles, tutorials, etc.), as well as a description of how AI was used — specifying for which tasks and which parts of the activity.

➠ **Additional sections may be required depending on the activity** (e.g., usage examples, feature list, technical choices, etc.).

Any required additions will be explicitly listed below.

- A detailed explanation and justification of the chosen algorithm and data structure must also be included.

English is recommended; alternatively, you may use the main language of your campus.

Chapter 6

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.

Once this assignment is passed, you will be allowed to add your `ft_printf()` to your `libft` so you can use it in your `C` projects at 42.

During the review, a brief **modification of the activity** may occasionally be requested. This could involve a minor behavior change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every activity**, you must be prepared for it if it is mentioned in the review guidelines.

This step is meant to verify your actual understanding of a specific part of the activity. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific timeframe is defined as part of the review.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **review guidelines** and may vary from one review to another for the same activity.

© 2025 Association 42. - All rights reserved