Structured Development

sd05

Summary:

In this module you will apply everything you have learned about structured development

Version: 4.2

Contents

Chapter I

Preamble

Hello, and welcome to your first Structured Development workshop!

This workshop is part of a mini-course for *42* students curated by *Prof. Enrico Vicario* and *Dr. Nicolò Pollini.* Throughout this course, you'll be introduced to a programming methodology that originated in the 1960s and has since evolved in tandem with the C language.

Structured Development is built on three key pillars:

- **Structured Programming**: This concerns the act of writing code in a clean, readable, and maintainable way. It emphasizes avoiding common pitfalls and code smells that lead to bugs and wasted time, especially during debugging.

- **Structured Design**: Good software design means organizing your codebase so that each function has a clear, well-defined purpose. The goal is to prevent situations where small changes ripple through the entire system, forcing you to rewrite large portions of code (yes, we're looking at you, *spaghetti code*).

- **Structured Analysis**: This is the most abstract level. It involves designing the architecture of an entire software system, especially when highly complex, based on its requirements and the flow of information needed to fulfill them.

Thus, **Structured Development** is the disciplined progression from problem to implementation: it connects *Structured Analysis*, *Structured Design*, and *Structured Programming* into a coherent workflow. Each phase refines the previous one (i.e., transforming requirements into data flow processes, then modular designs, and finally into reliable, maintainable code), ensuring clarity, traceability, and logical structure throughout the software lifecycle.

> **Remember:** Code is not just written to work, it's written to be used. If it's worth using, it's worth writing for someone else to read, understand, and maintain. You never write code *just* for yourself.

Chapter II

Introduction

Until today we've explored three different software development methodologies, but how are they actually applied in a real project?

If you've found yourself asking this question, you're not alone. For educational purposes, we have walked through the process in reverse to keep things clean and well-separated (just like your code should be).

It's now time to piece everything together. Structured Development is about approaching a problem methodically:

- First, by **defining the software requirements and data flows** through *Structured Analysis*, to create a clear and comprehensive project plan that outlines all the different processes and responsibilities.

- Next, **each process is refined into a logical hierarchy of modules** using *Structured Design*, keeping the data cohesive and the logic decoupled.

- Finally, **every module is implemented** using the principles of *Structured Programming*, improving readability and enabling axiomatic reasoning.

In the **June 20** lecture, you'll receive more in-depth information about Structured Development.
approach the projects with an open mind and see how it goes. At the end of the course, you-ll receive feedback on *both* versions. You're encouraged to explore freely: discuss ideas with your classmates, search for insights online, and feel free to use AI tools if they help you reason better.

However, a word of caution about **Large Language Models (LLMs)** like ChatGPT, Gemini, and similar tools:
Researchers (including us) have studied the impact of over-relying on AI in learning-driven development, and the findings suggest that, while LLMs can boost your short-term performance, **overusing them can hinder your long-term growth**. The knowledge and skills you build now are what your future success depends on.
Here's our recommended approach:

- **Start by solving the problem on your own.**

- **If you get stuck, ask your peers or look for help online.**

- **If you're still stuck, and no one can help, use LLMs to understand *how* to approach the problem**, but don't let them solve it for you, unless you're certain that solving this particular problem is not important to your learning journey.

Chapter III

General instructions

- This document assigns you **one mandatory** and **one optional** full-sized C programming project. We believe you won't have time to complete any more than that, but you're welcome to prove us wrong!


```
the evaluation will prioritize quality over quantity.
```

This marks the final phase of the mini-course, but you'll still have time to work on your project(s) after the lecture. So stay focused and *do your best*, just don't stress yourself trying to finish everything in a single day.

Chapter IV

Evaluation criteria

Unlike in previous experiences, you will be evaluated not only on whether your solution works and meets the requirements, but also on the **reasoning behind your choices**. In fact, this is the primary focus of the evaluation. *(No pressure, though)*

Here are a few important notes:

- **Your code must compile.** Minor oversights or non-critical errors may be tolerated, as long as the core functionality is preserved.

- **You're not required to follow the 42 Norm**, but it's a good starting point if you're unsure about what constitutes clean and readable code.

- **You don't need to code everything**, if you feel stuck on a difficult function and you are close to the deadline, just mock it and move on.

> To `mock a function` means to create a simplified version of it, often with hardcoded or simulated behavior. Instead of executing the real logic, a mocked function can return predefined or randomized outputs of the correct type, allowing you to test how other parts of your program interact with it before developing its actual implementation.

> **Note:** This is the `first version` of this document, so it may contain errors or inaccuracies. If you spot any issues, please let the staff know as soon as possible. Thank you!

Chapter V

Project 00

	Exercise 00
	Project 00
Turn-in directory : *ex*00/	
Files to turn in : `*.h, *.c, Makefile`	
Allowed functions : `free, malloc, get_next_line, ft_printf` or any equivalent you coded, all the functions present in your Libft	

- Choose any of the exercises from 'sd04' and implement it.

- Good luck, and have fun!

> Never underestimate the importance of the analysis and design phases: neglecting them can lead to serious issues for the developer who picks up the project. **Especially if that developer is you**

Chapter VI

Project 01 (bonus)

	Exercise 01
	Project 01

Turn-in directory : *ex*01/
Files to turn in : `*.h, *.c, Makefile`
Allowed functions : `free, malloc, get_next_line, ft_printf` or any equivalent you coded, `all the functions present in your Libft`

- Choose any of the bonus exercises from 'sd04' (ex04-ex07) and implement it.

- You may choose the bonus version of the exercise you selected for pj00.

- Good luck, and have fun!

Chapter VII

Project 02

	Exercise 02
	You really want more?
Turn-in directory : *ex02/*	
Files to turn in : `*.h`, `*.c`, `Makefile`	
Allowed functions : `free`, `malloc`, `get_next_line`, `ft_printf` or any equivalent you coded, all the functions present in your Libft	

Why are you still here? Just to suffer?

Chapter VIII

Submission and peer evaluation

Create a personal repository named:
    42xunifi-structured-development-2025-<your-intra-login>
Share it with the staff and organize the exercises as follows:

```
sd00/
 ex00/
 ex01/
 ex02/
 ...
sd01/
 ex00/
 ex01/
 ...
sd05/
 pj00/
 pj01/
 ...
```

You can share your repository by filling out this form (Only if you haven't done so already)

- **Update your repository regularly**, and make sure to **push all completed exercises before the deadline**.

- **This workshop's deadline is June 20**. Any changes made after this date **will not be considered** for assessment purposes.

- This exercise follows the **42 methodology**, so **peer-to-peer collaboration is allowed and strongly encouraged**.

- **Formal peer evaluations via Intra will not be available**, but you are still welcome to ask your peers for feedback on your work.

Our feedback won't be immediate, and unfortunately we've reached a level of abstraction so high that it's difficult to even think of an automated tool capable of providing meaningful feedback on something so layered and conceptual. Tools like **SonarQube** and **CCCC** still apply, but they operate at a lower level of abstraction. To help you track your progress, we recommend using AI tools (i.e., any LLM of your choice) to get feedback on your design ideas. For example:

- *"In the context of Structured Analysis (as formalized by Tom De Marco in 1978), I was given the following problem statement.*

- *I was thinking of structuring the project like this... and then that...*

- *What do you think?*

- *Here's the problem statement I was given: <insert-problem-statement-here>."*

We can't stress this enough:

- **Don't let AI dictate your actions**, always start by thinking through a solution on your own, then use AI for feedback and refinement.

- **Keep things simple**, you're here to *learn* how to engineer software, no one expects you to *be* a software engineer after just one week.

- **Don't fall down the rabbit hole**, if something gets too complex, move on to the next task. Aim to produce at least *something* for each exercise.

Chapter IX

Setting Up SonarQube (SonarCloud) with GitHub

1. **Create a public GitHub repository** for the course, named:
   `42xunifi-structured-development-2025-<your-intra-login>`.

2. Go to SonarCloud and click **"Start now."**

3. Sign up using your **GitHub account**.

4. When prompted, **import an organization**. You can use `"42"` or your GitHub username.

5. **Authorize access only to selected repositories**, and choose the one you created in step 1.

6. Choose a **Name** and a **Key** for your organization (your username is usually fine for both).

7. Select the **Free Plan** when prompted.

8. Click **"Create Organization."**

9. Select the previously created repository and click **"Set Up."**

10. For code technology, choose **"Previous version."**

11. Click **"Create Project."**

12. *Profit.*