Linked List

Requirements Definition

March 2025

Linked List

Partigiano Pratice
Greece

Συγγραφέας:

Partigiano.

Client:

Unknown Organization

March 2025

Table Of Contents

Table of References

Chapter 1: Version History

1.1 History Table

Date	Version	Description	Author
20/3/2025	<1.0>	1st version of the requirements definition document	Partigiano

Chapter 2: Introduction

2.1 Objective

The objective of this project is to develop a **robust**, **one-way *linked list***.

Chapter 3: Development process and organization issues

3.1 The Aproach

The development team will implement the “Scrum” approach to put through this project, i.e., plan a number of sprints during which the team shall implement **user stories** from the project backlog and their **tests**. The deadline for the project is 30/3/2025.

3.2 Issues

--

Chapter 4: Functional Requirements / User Stories

4.1 General User Stories

US ID	User story
US1	As a user, I want to be able to **create** my linked list.
US2	As a user, I want to be able to **add nodes** to my linked list, either in **start** or **finish** or **at index** ‘n’.
US3	As a user, I want to be able to be flexible, in the sense of the data that my nodes will hold.
US4	As a user, I want to be able to **delete** a node of my linked list either in **start** or **finish** or **at index** ‘n’.
US5	As a user, I want to be able to **retrieve** the **data** that a node holds.

Chapter 5: Non-Functional Requirements

[NF1] Maintainability: In software engineering, maintainability is the degree of effectiveness and efficiency with which a product or system can be modifiend by the maintainers. In the case of this project, we specifically focus on the following concerns:

- **[NF1.1] File-separation**. Each file has a specific role/purpose and **does not account** for how or what other files are doing, as long as, their interface is being met.
- **[NF1.2] Single-job functions**. Each function of each file has one specific job to do.
- **[NF1.3] Function Override**. Support function override to being able to **execute functions in an order that may not be known at compile time** AND without doing so without using conditional statements. This will be done via **Function Pointers**. This is particularly useful to avoid compiler screaming at you for multiple function definitions, and doing so with function pointers will contribute to project's maintainability. **Drawbacks**: **No** good **pipeline** and **branch predictions**. Such is life.

[NF2] Usability: In software engineering, usability concerns the ease of use and learnability. In the context of this project, the software is written is such way. That is should be **easy** for the fellow programmer **to use**, via **.h files only**. Furthermore, if he/she wants to learn **more details** about it, the **.c files** should also be a piece of cake to follow, with a smirk on the face.

Chapter6:Technical Requirements/Constraints/Recomendations

6.1 Technologies

Following, there is a list of technologies that Object Army uses to develop its clean software:

- C
- Clion
- CUnit
- Github