Matej Dedina

A C Data Structure Book

BOOK I

February 10, 2026

Springer Nature

Preface

This book describes the process of designing and implementing a personal data structure library in the C programming language. Each implementation is centered around two types of implementations - an infinitely expandable structure able to hold any amount of elements, and a finite one able to hold only a certain maximum amount.

The goal of this work is not to be the go-to data structure book for C, on the contrary, the name is meant to represent an affordable book for the public which doesn't take itself seriously.

The book will be divided into parts classifying all structures into categories of chapters. Each chapter will explain the process of creating said structure and the code used to make it.

Since the code is in C a decent understanding of the programming language is required, that primarily includes - structures (struct), pointers, memory allocation, and last but certainly not least - function pointers.

All the source code will be available at github under the `Unlicense License`.

If I have to put a disclaimer I just want to clarify that the reason why some paragraphs may be written like this is because the book was created using LaTeXin Visual Studio Code and I hate seeing `Overfull \hbox` highlights.

Contents

Part I
Restricted sequential structures

The name of the first part comes from the inability to find a categoric name for only stacks, queues and deques. Online sources categorize those as linear dynamic data structures, the problem is... linked lists are also part of this category. I want to put list implementations into a separate book part, thus I hereby coin the term `Restricted sequential data structures` as a subcategory ob dynamic linear data structures. I know nobody will take this declaration seriously, but for the sake of this book please bear with it.

Chapter 1
Stacks

Lets imagine we have a stack of books. The normal way we can add another one to the stack is by putting it at the top, if we put the book with the cover pointing upwards one is able to know what book exactly is at the top. To remove a book we just grab the top one and put it somewhere else.

By making space for another stack of books and adding all the books from the first one while the constrains of putting and removing still remain we get a new stack, but with a specific property that, when compared with our initial one, makes us tilt our head (literally), because the new one is just like the last one, but with the books in opposite order. This little property allows us to invert the order of any linear data structure without knowing how it's implemented, if the structure allows us to add and remove elements.

Continuing with the two stack analogy, what if instead of books we stored a game of chess. By adding these moves one-by-one 1.1: E2-E4, E7-E5, G1-F3, B8-C6, D2-D4, E5-D4, F3-D4, F8-C5, C2-C3, D8-F6, D4-C6, F6-F2; we get probably the funniest chess game in modern history [1]. And on top of that if we remove the two moves from the top and reset the pieces (F6-F2 becomes F2-F6), iteratively it is possible to return to the initial state of the game. And to top it all of, as probably every data structure book and article about stacks writes this mechanism is used by search engines when you want to return to the previous pages and back.

Stacks are also used to store function calls in programming languages like C to be able to know to what previous in memory place the program should go to after the top function returns.

1.1 Implementations of a Stack

Every computer science student learns that the two most common ways to implement a stack in C are:

1. Singly linked lists

[1] that can be watched via `https://www.youtube.com/watch?v=e91M0XLX7Jw`

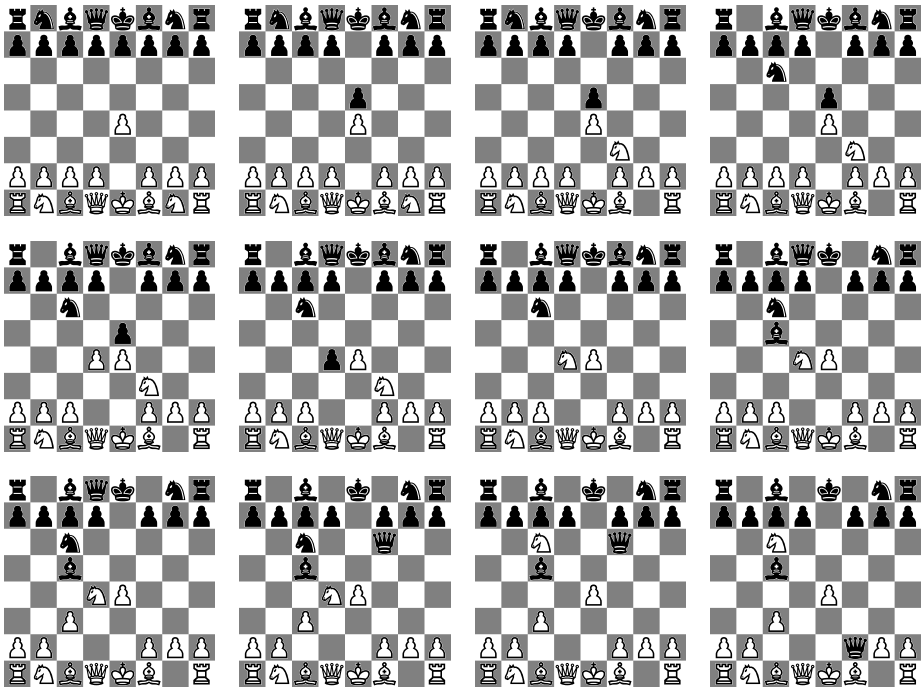

Fig. 1.1 The entire game of chess between content creators xQc and MoistCr1tikal.

2. Static/dynamic arrays

So why does this book have three subsections for stack implementations?

1.1.1 The humble linked list

For a more in-depth explanation of lists you can infer to PART II Linked lists, but in short a linked list is a linear data structure which allows us to store data sequentially as well as store a reference to the next data item 1.2.

The linked list consists of two special nodes which represent the beginning and the end - head and tail. Most linked list implementation will store the reference to the head, so it is generally easy to access the first element as oppose to the tail, since the user needs to follow all the arrows until they reach the tail.

1.2 Dynamic Array Stack Structure

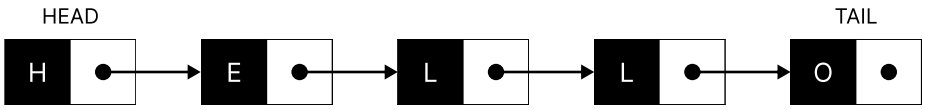

Fig. 1.2 A simple graphical linked list example.

Part II
Linked lists

Appendix A
Chapter Heading

Use the template *appendix.tex* together with the Springer Nature document class `SNmono` (monograph-type books) or `SNmult` (edited books) to style the appendix of your book.

A.1 Section Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations.

A.1.1 Subsection Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. A.1.

For multiline equations we recommend to use the `align` environment.

$$\mathbf{a} \times \mathbf{b} = \mathbf{c}$$
$$\mathbf{a} \times \mathbf{b} = \mathbf{c} \tag{A.1}$$

A.1.1.1 Subsubsection Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. A.1.1.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

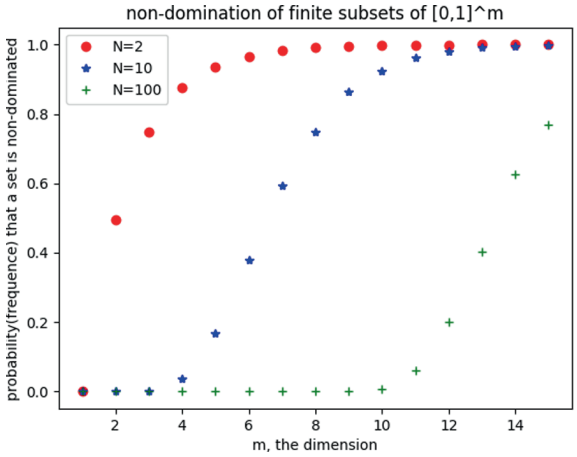

Fig. A.1 Please write your figure caption here

Table A.1 Please write your table caption here

Classes	Subclass	Length	Action Mechanism
Translation	mRNA[a]	22 (19–25)	Translation repression, mRNA cleavage
Translation	mRNA cleavage	21	mRNA cleavage
Translation	mRNA	21–22	mRNA cleavage
Translation	mRNA	24–26	Histone and DNA Modification

[a] Table foot note (with superscript)

Glossary

Use the template *glossary.tex* together with the Springer Nature document class SVMono (monograph-type books) or SVMult (edited books) to style your glossary in the Springer Nature layout.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

Solutions

Problems of Chapter ??

?? The solution is revealed here.

?? Problem Heading
(a) The solution of first part is revealed here.
(b) The solution of second part is revealed here.

Index