Modern C

Modern C takes a very formal and bottom-up approach, aiming to equip programmers with a fundamental understanding of how C works and the language to describe it.

Good for readers who like:

- knowing the vocabulary and C jargon to describe every part of their code
- a "from the ground up" approach, spending extra time on the low-level behavior
- a minimal textbook aesthetic
- data structures and algorithms, calculus, and linear algebra, and are excited to challenge themselves by connecting those concepts to C programming

Takeaway 0.1.1.1 *C is an imperative programming language.*

Takeaways are used to highlight important concepts.

[Exs 6] Correct listing 1.2 step by step. Start from the first diagnostic line, fix the code that is mentioned there, recompile, and so on, until you have a flawless program.
[Exs 7] There is a third difference between the two programs that we didn't mention yet. Find it.

Exercises are somewhat hidden as footnotes; you need to be attentive of footnotes as you read this book. This may be unintuitive for some readers.

This section covers

- C grammar
- Declaring identifiers
- Defining objects
- Instructing the compiler with statements

Chapters are nicely bookended with an overview of the content and a summary at the end.

Every level of understanding has its own mascot: this magpie to the right is for level 0: encounter.

Exercises are hidden in footnotes for concept retention. Mini-project-sized challenges break up major topics (many require fundamental data structures and algorithms/calculus foundations).

21st Century C

In 21st Century C, content is presented in an irregular structure, working to introduce best practices first. It is the least code-dense textbook, and focuses on covering what other textbooks do not.

Good for readers who like:

The Fault Is in Our Stars
A funny section header.

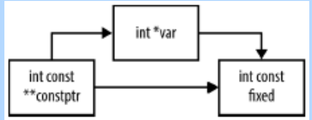
- a bit of humor in their textbook
*- to do things the best way possible *the first time**
- song references
- to focus on writing code that is readable and has a friendly UI
- jumping right in to best practices in modern C as they have prior experience programming in the language
- a book to share information for the sake of it being useful, not necessarily because all other textbooks cover it

	Static	Auto	Manual
Array size can be set at runtime		◊	◊
Can be resized			◊
Jesus weeps			◊

Tables, like on the left, which compares different types of memory, are used to highlight concepts throughout the book.

Callouts are used to highlight exercises for the reader.

Your Turn: Go through your programs and delete the `return 0` line from the end of `main`; see if it makes any difference.



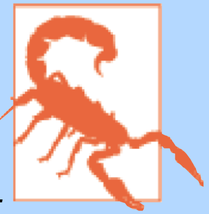
Callouts are also used to block out diagrams that aim to visually reinforce concepts related to code flow, among other things.

Chapters 7 and 8 make this textbook stand out from the others. Regardless of which textbook you choose, these chapters are a worthwhile skim.

Important C Syntax that Textbooks Often Do Not Cover

Inessential C Syntax that Textbooks Spend a Lot of Time Covering

The few exercises in the book serve to help reinforce content immediately presented. They're fairly bite-sized, intended to provide some practice before continuing the chapter.



The scorpion is used to warn the reader of something.

Head First C

In Head First C, content is written in a casual, narrative format, with lots of space for active participation.

One of many silly visuals to introduce or relate to a new concept.

Good for readers who like:

- writing directly in workbooks
- visual aides, often comical or sarcastic
- corny and somewhat outdated humor
- when examples build on themselves throughout a chapter instead of being one-off code snippets
- when a scenario is presented that the reader might come across in their code, and then the new content is presented as the solution to that scenario

Thought bubbles like the one to the right are often used to suggest a transition to a new concept.

Hello? I really don't **care** how the C language solves the problem. Just put the functions in the **correct freaking order**!

Common section breaks include these sections to summarize, clarify, and warn the reader.

C functions like printf() and scanf() use the Standard Output and Standard Input to communicate.

Summaries are presented as a collection of post-its at the end of every chapter.

Exercise solutions are presented immediately after the exercise. The "handwritten" note on this example is a common feature throughout the book.

```
int main()
{
  char search_for[80];
  printf("Search for: ");
  scanf(search_for, 80, stdin);
  find_track(search_for);
  return 0;
}
```

This version is using scanf() and would allow the user to enter 81 characters into the array.

Exercises are embedded within the chapter - some take up several pages at a time, and typically the outcome of the exercise serves as a transition to new content. They are presented in a workbook-style format.

more formal ← → *less formal*