cntlib

```
 ██████╗███╗   ██╗████████╗██╗     ██╗██████╗ 
██╔════╝████╗  ██║╚══██╔══╝██║     ██║██╔══██╗
██║     ██╔██╗ ██║   ██║   ██║     ██║██████╔╝
██║     ██║╚██╗██║   ██║   ██║     ██║██╔══██╗
╚██████╗██║ ╚████║   ██║   ███████╗██║██████╔╝
 ╚═════╝╚═╝  ╚═══╝   ╚═╝   ╚══════╝╚═╝╚═════╝ 
```

The preface

WHAT

This library provides a basic object-oriented (OOP) style API for pointer management. It also offers you abstractions and ideas to help you create fast, well-architected code. The philosophy of this library is to write a lot of code, to write code quickly, and to write readable code in the C language. The disadvantage of this library is the sacrifice of performance in favor of a template, so that you don't have to think too much. The idea of this library is to forgo good algorithms in order to save time.

WHY

I haven't written that much code in C, but I've constantly struggled with a few things:

- I wanted to create a massive architecture.
- I wanted to create a flexible architecture.
- I constantly had to think and spend time on it.
- Recurring issues in pointer algorithms.
- I had to write a lot of code.

I bought the book "C Programming. Mastery. Principles, Practices and Patterns", but it mostly talked about good code. Good code doesn't solve the pointer problem. And what names should I give to pointers? After all, we use them differently each time. In any case, I didn't want to bother with algorithms, and at the same time, I wanted to write good code. And most of the time, or rather, all of the time, I write code alone. I can't find my minions who would do the dirty work for me. So, this pain has tormented me for a year now.

The last straw was the FAT32 file system driver, where I constantly worked with pointers, and the code was hard to read. In addition, I was faced with byte-ending and tail-chaining problems. I still haven't found any minions, and I will not rewrite code for anything. I want to gain complete control over the computer, and refactoring is a waste of time. I wanted code that wouldn't need refactoring. Observing my pointer algorithms, I saw something in them. I saw that pointers in algorithms are counters, and counters are objects. Then I realized that pointers are objects. I knew about OOP, but I didn't know how to apply it in C, or rather, I didn't understand where to apply it. We work with registers, with bits, with bytes, not with objects. Creating a bit object is too cumbersome for C. We work with transistors, fans, and interfaces. There's nowhere to apply OOP, it's very, very cumbersome.

HOWTO

You can create simple examples that demonstrate the library's capabilities and its underlying concepts. Also, after building the project, you can also generate a book in PDF format.

```
git clone https://github.com/orca-li/cntlib.git
make
```