Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.30 Smart Pointers in Rust - Box

Unlike reference which just borrow value from other types, smart pointer on the other hand own values to which they have reference.

Most commanly used smart pointer is `Box<T>`

Box allocates data in heap and maintains a pointer in stack to reference that data.

```
fn main() {

    let b: Box<i32> = Box::new(5);

    println!("Value stored on heap is {}", b);

}
```

In the above program, rust allocates memory for value - 5 in heap and maintains a reference to that value in stack.

This smart pointer implements `Deref` trait and `Drop` trait implicitly. `Deref` trait makes `Box<T>` to act as reference and `Drop` trait is used to do the cleanup of heap and stack when the `Box<T>` goes out of scope.

For example, consider a data structure - `D` with data 1 GB. If we want to move data of `D` to another data structure `D1` , we tend to think of copying data from `D` to `D1` and it takes time as data increases.

But, if we incorporate usage of `Box<T>` in this usecase, we don't have copy entire data, we just have to transfer reference in stack from one owner to another owner.