Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.25 C & Rust - Constants & Replacements

Rust has this default behaviour of making any variable without `mut` before variable name as immutable. So, does C have this behaviour?

Yeah. But, it has its own way of dealing things. C doesn't have rust's default behaviour. Unlike rust, C makes each variable as mutable when they are declared. But, we can make them immutable by using something like `const` keyword.

Oh.. wait a second.. I guess, rust does have this `const` keyword. Then, how C's `const` is different from rust's `const` .

Rust's `const` can be compared with C's macro `#define` slightly (but not completely).

What's this `#define` ?

C has set of preprocessing directives. These are sometimes dubbed as `macros` . During preprocessing stage in C (happens before compilation step), C uses these preprocessing directives to perform some actions on code logic in program file. For example, when we use `#include<stdio.h>` in program file, `#include` makes C to get the contents of the file which is referred by this `#include` directive. In this case, C gets contents of `stdio.h` file and places that info into program file before compilation starts. So, coming to this `#define` , it contains two parts. The first part is like label and second part is what we need to replace the label with. For example, when we write `#define PI 3.14` , C checks the entire program to see usage of PI. As soon as C encouters `PI` in program logic, it just replaces `PI` with `3.14` . This whole thing happens before compilation.

Oh... lot of stuff is going on in C.

Yeah.. same process happens with rust usage of `const` but during compilation step (Rust doesn't have `preprocessing` step, I guess). Rust's `const` is used with types but C's `#define` is used to replace some part of code whether that code is related to types, expresssions (`#define add(x,y) x+y`) etc., And, C and rust are not tinikering with memory in this process. They are just replacing some code in program before the start of `run-time` of program during which variable's get memory from RAM.

```
#include<stdio.h>
```

```c
int
main() {

    /*
    * Any reassignment of PI after this line
    * results in an error
    */

    const float PI = 3.14;

    printf("%.2f", PI);
}
```

```c
#include<stdio.h>
#define PI 3.14
#define add(x,y) (x+y)

int
main() {

    printf("%.2f\n", PI);
    // add(x,y) is replaced with (x+y)
    // So, `printf` looks like printf("%d\n", (1+2));
    printf("%d", add(1,2));
}
```

```rust
fn main() {

    const PI:f32 = 3.14;
    println!("{}", PI);
}
```