Hack `postgres` Source Code: Vol I

Chapter 1 : C & Rust

1.2 Printing First Variable Declaration in `main.c` in `postgres`

"So, what can we do with C & Rust other than showing some text on console screen" Go asked showing her enthusiasm.

She is the only student who opt this summer course of "Hack `postgres` Source Code" in her Masters.

"Umm, we can do simple math" Pi said in his appealing voice.

"Well, can you tell me how to do that" Go asked.

"yeah, sure"

Pi recollected some information about memory in computers vaguely and tried to convey that to Go.

"So, to begin with, computers have two categories of memory . Those who pioneered in that memory field named one of them as Random Access Memory and other as permanenet memory. Random Access Memory which we can also call it as RAM is volatile memory. This memory isn't meant to last long in computer and wiped off after computer is shut down. Unlike this memory, permanent memory comes either in HDD also called as Hard Drive Disk or in SSD also known as Solid State Drive. This type of memory is permanent and shutting down the computer doesn't make the data in that memory cleared off"

"Why are you telling me about memory stuff?" Go asked coolly.

"I am just setting the background before getting into the details of doing math in C & Rust" Pi said in calm and unhurried voice.

"Ok" Go Said.

"Coming to math thing where you are eager to know, we can't directly do math in C & Rust. C & Rust should have some means to perform math in computer."

"When we do math in our notebook, we deal with negative number, positive numbers, fractional numbers. So, in C & Rust, they are categorized those numbers under `integers` and `floats` . `floats` wrap all fractional numbers and non-fractional numbers are placed under `integers` . In computer, we need to store numbers in some location in memory which is usually RAM before we perform math on them using our C/Rust. But, how much of our RAM memory should we need to allocate to those numbers?" asked Pi casually.

"What about some random size of memory" said Go in low pitch voice.

"If we assign random size of memory then there will be wastage of memory. Because of this, in C & Rust, we fix memory size for each category of numbers based on their size. So, integers and fractional numbers are divided into some subcategories to conveniently deal with small, large, postive only, postive/negative integers, low precission fractional number and high precission fractional numbers"

"Ok" said Go with high sigh.

Pi felt that he was pushing lot of theoretical stuff into Go's mind and wanted to show her that number categorization in C & Rust.

"So, if we want to allocate 1 byte of memory for integers, we use `i8` as type in Rust. If we are sure that we use that 1 byte for only positive integers, we use `u8` as type. In the same way, we use `i16` for 2 bytes of memory, if we use it for only positive integers, we use `u16` . This goes for 4 bytes, 8 bytes and 16 bytes in Rust."

"What if I want to store only negative numbers, do we have any type for that?" asked Go interrupting Pi.

"There is no such type. It is because we accormodate integers with sign whether +/- using `i` prefixed types and those without signs are handled by the types that are prefixed with `u` ." said Pi as he was referring a definition in some book.

"And in C, we use `short int` as type to allot 2 bytes, `int` for 4 bytes `long int` for 8 bytes. We just add `unsigned` before them if we want to use those bytes only for positive numbers."

"More numbers of bytes helps in holding large number of integers"

"Don't we have any type to represent 1 byte in C" asked Go showing curiosity.

"No", said Pi.

"In the same way, we deal with fractional numbers in C and Rust. If we want to allocate 4 bytes for fractional numbers or floating point numbers or simply floats, we use `f32` as type. And also, we can allot 8 bytes for floats with type `f64` . We get more precission of floats if we use more number of bytes. In C, we use the type `float` to hold 4 bytes and `double` to hold 8 bytes." Said Go in fast paced manner.

"What about unsgined floats? do we have them?" asked Go innocently.

"As per IEEE754 specification, there is no such thing" said Pi who also didn't know about exact reason.

"Ok" said Go with an intention to leave that in the air as she was not interested to know about that IEEE stuff.

"So, how can we actually use those types to perform math in our computer through this C or Rust" asked Go.

Pi was about to write a sample code for Go while she was asking.

"I will demonstrate that through some sample code" said Pi.

Pi started to type sample code in his computer which was connected to a projector to show his code on the white wall infront of the Pi who was sitting comfortably leaning in the chair.

On Projector -

```
// C Code

#include<stdio.h>

int
main() {

    /*
    * Wanted to set 4 bytes for my integers
    */
    int a = 1;
    int b = 2;

    printf("%d", (a+b));

    return 0;
}



// Rust code

fn main() {

    /*
    * Wanted to set 4 bytes for my integers
    */
    let x: i32 = 1;
    let y: i32 = 2;

    println!("{}", (x+y));
    }
```

"you can play with different types in the same way" Pi told Go after writing the code in his computer.

"Ok" said Go.

"I want to know, how we represent something like right or wrong, in other word true or false kind of things in C/Rust?"

"We have types for that as well in C & Rust" Said Pi in his calm voice.

"Those types are usually called as boolean types. In Rust, we allocate one byte of memory with a type `bool` and same amount of memory is allocated with a type `bool` in C also. This `bool` is available in modern C standard which is called as C99 standard by including header file `stdbool.h`."

"Is there any new format specifier for `bool`" asked Go.

"We need to use existing format specifiers like `%d%` for `bool` as there is no information in C99 standard" replied Pi asif he got the information from stackoverflow website.

Go wanted to know about this type particulary as she wanted to replicate the first line in `postgres` source code `main()` first line.

She quickly opened up her `vim` editor and tried to write a program in C & Rust.

In Go's `vim` editor -

```
//C Code

#include<stdio.h>
#include<stdbool.h>

int
main() {

    bool do_check_root = true;

    printf("%d", do_check_root);

}


//Rust Code

fn main() {

    let do_check_root: bool = true;

    println!("{}", do_check_root);

}
```

Go wanted to know more about format specifiers in the rust and she didn't know what the hell was that `let` and `:` between name and type in variable declaration in rust. But, she didn't want to ask about them now as she felt little exhausted with the lot of information from Pi about types in C and Rust.

"I wanted to ask some doubts tomorrow" said Go in tired voice.

Pi understood Go's situation and didn't want to fill her mind anymore with C & Rust.

"Sure" said Pi.

They pushed their laptop into their bags and started to go out of the college. Go texted Pi to remind their dinner plan tonight. They were dating for 1 year since Pi got into her Masters.