1 Foundations

1.1 Memory

There are three main memory regions:

- The Stack
- The Heap
- Static memory

The **stack** contains the memory necessary for each function call and the variables that it uses, each time a function is called it gets a place on top of the stack and once it finishes the memory is de-allocated.

The **heap** contains memory that is not related with any function or any threat so it can be accessed at any time from any function. To use the heap the most common structure to use is the **Box<T>**.

The static memory contains the binary code for the program and the static values that are set at compile time and last all the life time of the program. The life time *static* indicates that a variable lives until the program end but doesn't need to be initialized at the start of the program.

1.2 Ownership

A *value* is owned by a *variable* that stores it on a memory region, when the value is assigned to an other variable or moved to an other region of memory (push to a vector, box it...) then the ownership is moved and the original variable no longer owns the value so it can no longer access it.

Some types are special and don't drop from the original variable instead are copied to the new one, so an exact copy is created on the corresponding memory region. This types have the **Copy** trait, primitive types like *i32*, *char*..., have this trait.

1.3 Borrowing and Lifetimes

You can borrow multiple shareable instances of a variable **&T** but just one that is a mutable reference to the variable **&mut T** and the compiler will make sure that when a mutable reference is alive no other reference to the variable is accessed.