

Student Name: Ya Zou

CS 381 - Homework 1

Goal: Research and explore a new programming language.

1) Select a programming language you have not used (NOT Haskell or Prolog). Give a brief history of this language. Who developed the language? When? Where?

Rust is a system programming language that was first created by software developer Graydon Hoare as a personal project on the Firefox browser while he was working at Mozilla Research in 2006, then Mozilla officially sponsored the project in 2009 and the first official version of the language, Rust0.1 was published in 2010. The goal of the language was to provide a more secure and efficient alternative to C and C++ for system programming. Rust is very similar to C and C++, which incorporates many of the keywords and commands from both languages. But it is not a direct clone, because it has some elements not found in either C or C++. [\[1\]](#)

2) Give a description of the language. What is its paradigm(s)? How is it categorized? Describe some of the features of the language. What is typing method/discipline?

Rust is multi-paradigms programming language and is an imperative programming language and it also follows functional programming paradigms which means rust allows you to write code in both functional and imperative styles, Rust is designed to be safe, concurrent, and fast. It has several paradigms, and some of the main paradigms of rust are memory safe, rust provides a number of features that help prevent common memory-related errors, such as buffer overflows and data races. Concurrency, rust has built-in support for concurrent programming through its lightweight “thread” model and message-passing concurrency. Object-oriented programming, Rust has a lot of features that support object-oriented programming. Such as structs, and enums. Algebraic data types: Rust supports algebraic data types such as enums and pattern matching, which makes it easy to define and work with

complex data types. Rust also has a unique ownership model that allows for safe and efficient memory management.

Features: First, rust is C-compatible: Rust is designed to be compatible with C, which means that some C libraries can be used in Rust. Second, Rust has a unique ownership model which allows for safe and efficient memory management. Third, Rust also supports functional programming languages. Fourth, Rust has a built-in package manager called cargo, which makes it easy to manage dependencies and share code between projects. Finally, Rust provides a strong and active community of developers and users who contribute to the language's development and ecosystem. These features make Rust a powerful programming language. **The typing method** of the Rust programming language is strongly and statically typed which means all types of variables must be known during compilation, and type cannot be changed at runtime[1]

3) Is the language compiled or interpreted? On what platforms is the language available? Is the language standardized? Are there different implementations?

Rust programming language is an ahead-of-time compiled language(compiled language), which means that it can be translated into machine code that is executed directly by a computer's CPU. Rust is available on a huge list of platforms such as Windows, macOS, Linux, and Unix. Rust is a standardized language which means that there is a specification that defines the language's syntax, semantics, and standard library. There are different implementations in Rust, for example, the official implementations rustc and rustup, there are also other non-official implementations like miri, and a Rust interpreter is used for testing purposes, however, the official implementation is the one that is aligned with the language specification.

4) Give examples of at least two control structures in the language (ie. for-loop or if-statement). Explain.

Example 1:

This is just a simple If statement, n equals 4 which is greater than 0, therefore the program below will print the message: “4 is positive” to the console.

```
let n = 4;
```

```
    if n > 0 {  
        println!("{}", is positive", n);  
    }
```

Example 2: if-else statement

In the program we assigned n equals 4 which is greater than 0, then the else statement is executed here, it will print the message “ 4 is positive” to the terminal.

```
let n = 4;
```

```
If n < 0{
```

```
    println!("{}", is negative", n);  
}else{
```

```
    println!("{}", is positive", n);  
}
```

Example 3: if let statement

The if let construct reads: “if `let` deconstructs `number` into `Some(i)`, then evaluate the block `{}`”.

```
let number = Some(7);  
if let Some(i) = number  
{  
    println!("Matched {:?}!", i);  
}
```

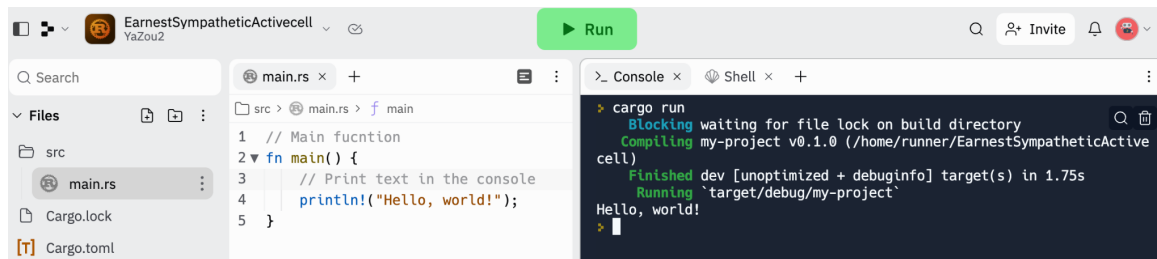
Code is referenced :[5]

5) To complete questions 6- 8 you will need to find an online compiler/interpreter to run code in your language, for example <https://onecompiler.com/> .

Alternatively, you can install the software or use the school’s server. What will you be using? Note: You can use the code you find online to answer questions 6-8 just include this information in the reference list for question 9.

I am going to use [replit](https://replit.com/) to compile programs.

6) Give a “Hello World” program written in the language. Describe how it works. Provide a screenshot of the execution of the program.



When this program is compiled and run, it will call the `main()` function, which in turn calls the `println!()` built-in function to print the message “Hello world!” on the console.

7) Give a program to compute the first `n` Fibonacci numbers where the user is prompted for `n` (if possible). Describe how the code works. Is the program iterative or recursive? Provide a screenshot of the execution of the program.

```
use std::io;
```

```
fn fib (n: i32) -> i32 {
    if n <= 0 {
        return 0;
    } else if n== 1{
        return 1;
    } else {
        return fib (n-1)  + fib(n-2);
    }
}
```

```
fn main() {
    println!("\nEnter a positive integer:");

    // get user nput
    let mut user_input = String::new();
    io::stdin()
```

```

        .read_line(&mut user_input)
        .expect("Failed to read line");

        //Convert string to integer
        let n =
user_input.trim().parse::<i32>().unwrap();
        for int in 0..n{
            println! ("fibonacci ({{}}) = {{}}", int,
fib(int));
        }
    }
}

```

Description of the code: The program starts in the main function and then calls the `println!()` to prompt a message to the user to enter a positive integer then store it in the variable `uer_input`, after that it converts `user_input` to an integer, then goes to the for loop to call the `fib()` function to generate the first n fibonacci numbers. For printing out the first 0 to n (n is user input) the program used a for loop which is iterative, but the function `fib()` is recursive.

Screenshot of the execution of the program:



```

> cargo run
  Compiling my-project v0.1.0 (/home/runner/FibonacciNumbers)
  Finished dev [unoptimized + debuginfo] target(s) in 1.08s
  Running `target/debug/my-project`

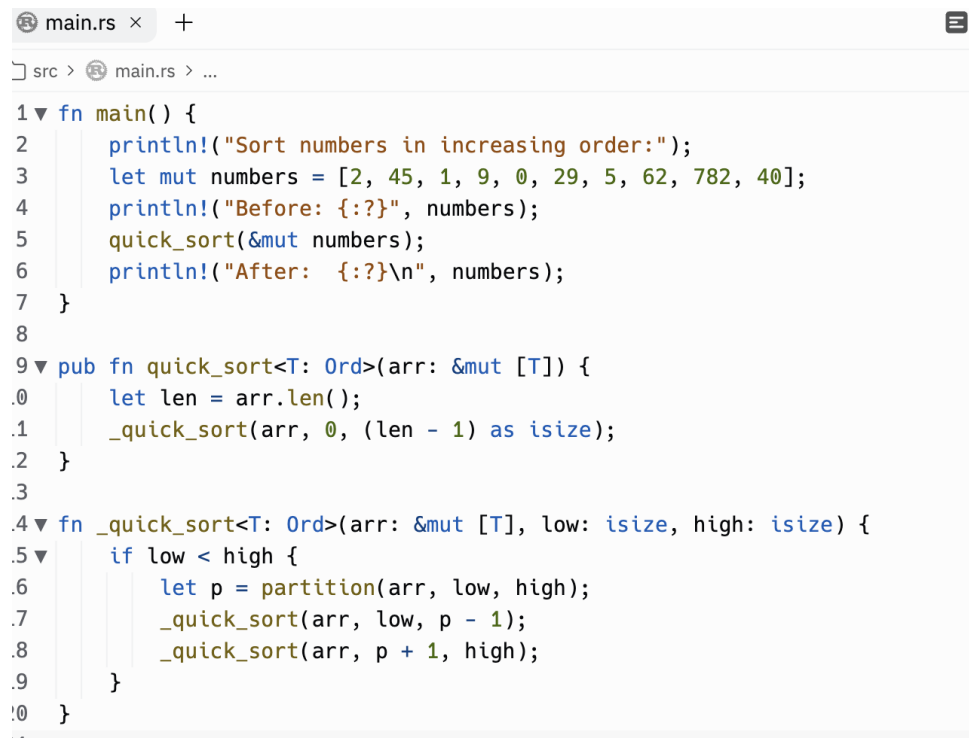
Enter a positive integer:
8
fibonacci (0) = 0
fibonacci (1) = 1
fibonacci (2) = 1
fibonacci (3) = 2
fibonacci (4) = 3
fibonacci (5) = 5
fibonacci (6) = 8
fibonacci (7) = 13
> 

```

8) Give a program to sort a list of integers. You can use any sorting algorithm. Describe how the code works. Provide a screenshot of the execution of the program.

Description of the code: The keyword `fn` is a function pointer to point to our code here, it is called just like a function, first it goes to the main function and calls the `println!()` to print the original integer array to the console, then it calls the `quick_sort()` which selects a pivot element then calls the `partition()` to split the original array into smaller elements on the left side and larger elements on the right side(two sub_array) until everything is sorted. Once all elements have been sorted according to the pivot(median value), the program will repeat for each sub_array.

Code is referenced in [\[4\]](#)



```
main.rs x +
src > main.rs > ...
1 ▼ fn main() {
2     println!("Sort numbers in increasing order:");
3     let mut numbers = [2, 45, 1, 9, 0, 29, 5, 62, 782, 40];
4     println!("Before: {:?}", numbers);
5     quick_sort(&mut numbers);
6     println!("After: {:?}\n", numbers);
7 }
8
9 ▼ pub fn quick_sort<T: Ord>(arr: &mut [T]) {
10     let len = arr.len();
11     _quick_sort(arr, 0, (len - 1) as isize);
12 }
13
14 ▼ fn _quick_sort<T: Ord>(arr: &mut [T], low: isize, high: isize) {
15     if low < high {
16         let p = partition(arr, low, high);
17         _quick_sort(arr, low, p - 1);
18         _quick_sort(arr, p + 1, high);
19     }
20 }
```

```

22 ▼ fn partition<T: Ord>(arr: &mut [T], low: usize, high: usize) -> usize {
23     let pivot = high as usize;
24     let mut store_index = low - 1;
25     let mut last_index = high;
26
27     loop {
28         store_index += 1;
29         while arr[store_index as usize] < arr[pivot] {
30             store_index += 1;
31         }
32         last_index -= 1;
33         while last_index >= 0 && arr[last_index as usize] > arr[pivot] {
34             last_index -= 1;
35         }
36         if store_index >= last_index {
37             break;
38         } else {
39             arr.swap(store_index as usize, last_index as usize);
40         }
41     }
42     arr.swap(store_index as usize, pivot as usize);
43     store_index
44 }

```

Screenshot of the execution of the program:

```

> cargo run
  Compiling my-project v0.1.0 (/home/runner/EarnestSympatheticActivecell)
  Finished dev [unoptimized + debuginfo] target(s) in 1.16s
  Running `target/debug/my-project`
Sort numbers in increasing order:
Before: [2, 45, 1, 9, 0, 29, 5, 62, 782, 40]
After:  [0, 1, 2, 5, 9, 29, 40, 45, 62, 782]

```

9) List at least three references you used for this assignment. Include any sites that you used to obtain code.

Cited resources:

[1][https://en.wikipedia.org/wiki/Rust_\(programming_language\)#Syntax_and_semantics](https://en.wikipedia.org/wiki/Rust_(programming_language)#Syntax_and_semantics)

[2]<https://doc.rust-lang.org/reference/items/implementations.html#:~:text=T here%20are%20two%20types%20of,trait%20implementations>

[3]<https://replit.com>

[4]<https://www.hackertouch.com/quick-sort-in-rust.html>

[5] https://doc.rust-lang.org/rust-by-example/flow_control/if_let.html

10) Would you like to learn more about this language? Would anticipate using this language in the future? Explain.

Yes, I would like to learn more about this language because I am going to study Rust programming language in my Operating System class. Because Rust language is a system programming language that is designed to be safe, concurrent, and fast. When I finish operating system II, I would like to try use Rust to implement an operating system because coding in Rust is as efficient as C or C++, but Rust guarantees of a modern, high-level programming language, also Rust is a powerful language that is well studied for a wide range of applications, and learning it can open up more opportunities for my career as a developer.