Writing the same program in 2 languages

To prove that programming has nothing to do with coding.

Index

About me

Timmy Ö, 25
aka: Vonriddarn

Started programming in 2013
Unity and C# through school.

Very average skill level
No guru, but good enough to make a point with this project.
Mainly use C# on and off for making projects.
No prior experience in Rust – never even seen rust code before this project.

Hobbies
Programming,
Cooking &
Photography.

The program

A translator for pirate language AKA ”Pippi Långstrump language”.
The language works by simply putting O's between consonants.

Examples:

Jag heter Timmy. → JoJagog hohetoteror ToTimommomy.

My name is Timmy. → MoMy nonamome isos ToTimommomy.

Program flow:

- Take user input
- Translate that input
- Output the translation of the input

How we'll do it (pseudo programming):

Cache vowels in a char array (UTF-8). ”aouåeiyäö”.
Initialize empty input variable. (string answer)

Wrap the entire program in a menu while loop. (Flag = exit or maybe use break;)
Greet user with text. (WriteLine)
Tell user to type .exit to close application.

Store user input in the *answer* string. (ReadLine)

(Early return pattern)
If *answer* == ”.exit” → Close application.
Else → Continue.

Create a tempOutput variable for the output.

Loop through the *answer* string.

If Character at position i is a vowel or is not a letter
    Do nothing and add to tempOutput.
Else
    If character is CAPITAL
        Add {CHARACTER}O{CHARACTER} to tempOutput.
    Else
        Add {character}o{character} to tempOutput.

When string is completely looped:
    Type out tempOutput to screen. (WriteLine)

Sourcecode C#

```
namespace PirateTranslatorCsharp;

class Program
{

  const string VOWELS = "aouåeiyäö";
  static string answer = String.Empty;

  static bool IsVowel(char c) => VOWELS.ToLower().Contains(Char.ToLower(c));

  static void Main(string[] args)
  {

    while(true)
    {
      Console.Clear();
      Console.WriteLine("Yarr ohoy matey! Welocme to me pirate translator!");
      Console.WriteLine("Do be typey '.exit' to leave me program! Yarr");
      Console.Write("Input: ");

      answer = Console.ReadLine()!;

      if(answer.Length > 0)
      {
        if(answer.ToLower() == ".exit")
        {
          break;
        }

        Console.WriteLine(Translate(answer));
      }
      else
      {
        Console.WriteLine("Oy, put something in me translator!");
      }

      // Pause the current itteration.
      Console.ReadLine();
    }
```

```
    Console.WriteLine("Good bye, matey!");
  }

  static string Translate(string textToTranslate)
  {
    string outPutString = String.Empty;

    foreach(char c in textToTranslate)
    {
      // Early return pattern
      if(IsVowel(c) || !Char.IsLetter(c))
      {
        outPutString += c;
        continue;
      }

      // This is a consonant!
      outPutString += Char.IsUpper(c) ? $"{c}O{c}" : $"{c}o{c}";
    }

    return outPutString;
  }
}
```

Time to finish: 15 minutes.

Sourcecode Rust

```rust
use std::io;

const VOWELS:&str = "aouåeiyäö";

fn is_vowel(c:char) -> bool
{
  return VOWELS.to_lowercase().contains(c);
}

fn main()
{
  let mut _answer:String = String::new();
  loop
  {
    print!("\x1B[2J\x1B[1;1H");

    println!("Yarr ohoy matey! Welocme to me pirate translator!");
    println!("Do be typey '.exit' to leave me program! Yarr");
    println!("Input: ");

    _answer = get_user_input();

    if _answer.trim_end().len() > 0
    {
      if _answer.to_lowercase().trim_end() == ".exit"
      {
        break;
      }

      let mut _out = translate(_answer);
      println!("{_out}");
    }
    else
    {
      println!("Oy, put something in me translator!");
    }

    // Pause the current itteration
    get_user_input();
  }

  println!("Good bye, matey!");

}

fn translate(string_to_translate: String) -> String
{
  let mut temp_output = String::new();
```

```
  for c in string_to_translate.chars()
  {
    if is_vowel(c) || !c.is_alphabetic()
    {
      temp_output.push(c);
      continue;
    }

    // This is not very nicely done, but I can't get rust to accept my characters
into strings.
    if c.is_uppercase()
    {
      temp_output.push(c);
      temp_output.push('O');
      temp_output.push(c);
    }
    else
    {
      temp_output.push(c);
      temp_output.push('o');
      temp_output.push(c);
    }
  }

  return temp_output;
}

fn get_user_input() -> String
{
  let mut _input:String = String::new();

  match io::stdin().read_line(&mut _input)
  {
    Ok(_n) =>
    {
      return _input;
    }
    Err(error) =>
    {
      println!("error: {error}");
      return String::new();
    }
  }
}
```

Time to finish: 2 hours.

What did I learn?

Language syntax vs logical programming

Programming is more about knowing patterns, data types and problem solving than actual language syntax.

A versitile programmer will get the job done no matter the tools given, albeit, with varying hits on performance and readability. It was obvious when creating this project that my lack of Rust knowledge was holding me back immensly, but it wasn't stopping me from learning and pushing through.

There was a lot of times where I was cursing out Rust as a language because it didn't work as I wanted it to, and I had to resolve issues in ways I wouldn't have chosen in a language I know better, such as C# or C++.

Even though it took me a lot longer, and the quality of the product is questionable I was able to complete the task I was given in a language I have never even seen before.

Time and cost

Although programming is not bound to a specific language you benefit a lot from specializing in one language or framework as you will be able to learn language specific optimizations and standards. You will also be more proficent when using a syntax you know by heart and don't have to look up every other method.

Due to my previous C# knowledge I was able to create a more proficient version of the program in much less than half the time it took me in Rust. This means that being specialized in languages is not a bad thing, as your proficency and speed increases the more you use it.

Going by this, what language should you learn if you're new?

Whichever language you want!

Although given the fact that data manipulation is one of the core concepts of logical programming, I would recommend a language that is *strictly typed*.
This will force you to learn this important aspect early on.

The real answer though is:
Whatever language you have fun programming in.

Learn one, learn all.