```rust
let mut number1: String = String::new();
let mut number2: String = String::new();
let mut choice: String = String::new();
```

En başta kullanıcıdan değer alınabilmesi için number1, number2 ve choice değişkenlerini başlattık.

```rust
println!("Please enter the first number...");
io::stdin().read_line(buf: &mut number1).expect(msg: "Giriş okuma hatası!");
let number1: f64 = number1.trim().parse().expect(msg: "Sayıya dönüştürme hatası!");
println!("Please enter the second number...");
io::stdin().read_line(buf: &mut number2).expect(msg: "Giriş okuma hatası!");
let number2: f64 = number2.trim().parse().expect(msg: "Sayıya dönüştürme hatası!");
```

Kullanıcıdan girdi alabilmek için io kütüphanesinden read_line() fonksiyonunu kullanıyoruz. Strin olarak aldığımız için ama bizim kesirli sayı olarak kaydetmemiz gerekiyor ondan parse() fonksiyonu ile çevirmemizi yapıyoruz. Ondan önce kullanıcıdan aldığımız girdinin önünde veya arkasında istenmeyen karakterler için trim() fonksiyonunu kullanıyoruz.

```rust
introduce();
io::stdin().read_line(buf: &mut choice).expect(msg: "Giriş okuma hatası!");
choice = choice.trim().to_string();
let mut result : Option<f64> = None;
```

introduce() fonksiyonu tamamen bilgilendirme amaçlı konsola yazılan bir fonksiyondur. Sonra kullanıcı hangi işlemi yapmak istiyorsa sayı girer ve gerekli işlem yapılır. İleride result değişkenine detaylı değineceğim.

```
if choice == "1"{
    result = calculate(Operation::Add(number1, number2));
}
else if choice == "2"{
    result = calculate(Operation::Substract(number1, number2));
}
else if choice == "3"{
    result = calculate(Operation::Multiply(number1, number2));
}
else if choice == "4" {
    result = calculate(Operation::Divide(number1, number2));
}
else if choice == "5"{
    break;
}
match result {
    Some(value: f64) => println!("Result : {}", value),
    None => println!("İnvalid transaction"),
}
```

Kullanıcı geçerli girdi girdikten sonra gerekli işlemler yapılıyor. result değişkeni ‘option’ tipinde değişken. Çünkü eğer kullanıcı geçersiz bir girdi girerse ileriki kod satırlarında ‘match’ yapısı ile kontrol edebillirim. Ondan daolayı calculate() fonksiyonumdan Option<f64> dönüyor. Böylelikle geçerli mi geçersiz mi işlem girilmiş kontrol etmiş oluyorum.

```
fn calculate(operation:Operation)->Option<f64>{
    match operation {
        Operation::Add(number1: f64, number2: f64) => {Some(number1 + number2)},
        Operation::Substract(number1: f64, number2: f64) => {Some(number1 - number2)},
        Operation::Multiply(number1: f64, number2: f64) => {Some(number1 * number2)},
        Operation::Divide(number1: f64, number2: f64) => {Some(number1 / number2)},
    }
}

fn introduce(){
    println!("1-Add\n2-Substract\n3-Multiply\n4-Divide\n5-Quit");
}
```