Rust – Vector & Tuple

Week 8

SCHOOL OF
SCIENCE AND
TECHNOLOGY
PAN-ATLANTIC UNIVERSITY

Rust - Vector

A vector is a container that stores the values like an array, but it has more advantages than an array data structure.

It can increase in size dynamically during runtime.

It is provided by the standard library that can store the value of any data type.

Its length defines the number of elements present in the vector.

Its capacity defines the actual allocated space on the heap of this vector.

Creating a vector in Rust:

Creating a vector in Rust

Practice 1: week-8/practice_1/src/main.rs

```rust
fn main() {

    // Using Vec::new()
    let v : Vec<i64> = Vec::new();

    // printing the size of vector
    println!("\nThe length of Vec::new is: {}",v.len());

    // Using macro
    let v = vec!["Grace","Effiong", "Basil","Kareem","Susan"];

    // printing the size of vector
    println!("\nThe length of vec macro is: { }",v.len());

}
```

Accessing a vector in Rust:

- Using subscript operator
- Using the get() method
- Iterating on the vector

Using subscript operator

Similar to the concept of indexing in other languages, subscript operators can be used to directly access the values in a vector through their index.

Practice 2: week-8/practice_2/src/main.rs

main.rs

```rust
fn main() {

    let v = vec!['C','O','M','P','U','T','E','R'];

    let mut input1 = String::new();

    println!("Enter an index value btw (0 - 7)");
    std::io::stdin().read_line(&mut input1).expect("Failed to read input");
    //index is the non negative value which is smaller than the size of the vector
    let index:usize = input1.trim().parse().expect("Invalid input");

    //getting value at given index value
    let ch: char = v[index];

    print!("{} is the character for index [{}]\n",ch, index);
}
```

Using get() method

The second way of accessing the vector elements is to use the get(index) method with the index of a vector passed as an argument.

Practice 3: week-8/practice_3/src/main.rs

main.rs

```rust
// Method to print the get value
fn value(n:Option<&char>)
{
    println!("Element of vector {:?}",n);

}

fn main() {

    let v = vec!['R','U','S','T','A','C','I','A','N' ];

    let mut input1 = String::new();
    println!("\nEnter an index value btw (0 - 8)");
    std::io::stdin().read_line(&mut input1).expect("Failed to read input");

    //index is the non negative value which is smaller than the size of the vector
    let index:usize = input1.trim().parse().expect("Invalid input");

    // getting value at given index value
    let ch: Option<&char> = v.get(index);
    value(ch);
}
```

Iterating on the vector

To access the vector we can also iterate through a vector-like we do in arrays. We can use **for loop** to iterate through a vector.

Practice 4: week-8/practice_4/src/main.rs

main.rs

```rust
fn main() {

    // Name vector
    let name = vec!["Mary","Sam","Sally","Greg","Ade","Mark","June","Ife"];

    // Age vector
    let age = vec![16,17,19,22,20,21,18,23];

    print!("\nAge allocation:\n");

    //loop to iterate elements in vector
    for i in 0..age.len()
    {
        // iterating through i on the vector
        print!("{} is {} years old\n",name[i],age[i]);

    }
}
```

Updating a vector

Practice 5: week-8/practice_5/src/main.rs

main.rs

```rust
fn main() {
    // Create an empty vector "City"
    let mut city : Vec<String> = Vec::new();
    // Print City Vector
    println!("The City vector has element {}",city.len());
    // Push new elements into
    let mut input1 = String::new();
    println!("How many Cities do you want to enter");
    std::io::stdin().read_line(&mut input1).expect("Failed to read input");
    let city_num:i32 = input1.trim().parse().expect("Invalid input");
    for count in 0..city_num {
        let mut input2 = String::new();
        println!("Enter City {}", count+1);
        std::io::stdin().read_line(&mut input2).expect("Failed to read input");
        let new_city:String = input2.trim().parse().expect("Invalid input");
        city.push(new_city);
    }
    print!("Your preferred cities are:\n");
    let mut count=1;
    // loop to iterate elements in vector
    for i in city
    {
        // iterating through i on the vector
        println!("{} {}", count, i);
        count+=1;
    }

}
```

Adding two vectors

Practice 6: week-8/practice_6/src/main.rs

main.rs

```rust
fn main() {

    // Create two vector
    let v = vec![1,2,3,4,5,6,7,8];
    let x = vec![5,6,7,8,9,10,11];

    // Use a for loop to add elements of the vector
    for index in 0..6 {
        let sum = v[index] + x[index];
        println!("{:?}", sum);
    }
}
```

Rust Tuple

Tuple is a compound data type. A scalar type can store only one type of data. For example, an i32 variable can store only a single integer value.

In compound types, we can store more than one value at a time and it can be of different types.

Tuples have a fixed length - once declared they cannot grow or shrink in size. The tuple index starts from 0.

The *println!("{}")* syntax cannot be used to display values in a tuple. This is because a tuple is a compound type. Use the *println!("{:?}")* syntax to print values in a tuple.

Initializing & Accessing Elements

Practice 7: week-8/practice_7/src/main.rs

main.rs

```rust
fn main() {
    // initialization of tuple with data type
    let datatype_tuple: (&str, f32, u8) = ("Rust", 3.14, 100);
    println!("Tuple contents = {:?}", tuple);

    // initialization of tuple without data type
    let no_datatype_tuple = ("Rust", "fun", 100);
    println!("Tuple contents = {:?}", tuple);

    // accessing tuple element at index 0
    println!("Value at Index 0 = {}", datatype_tuple.0);

    // accessing tuple element at index 1
    println!("Value at Index 1 = {}", datatype_tuple.1);

    // accessing tuple element at index 2
    println!("Value at Index 2 = {}", datatype_tuple.2);
}
```

Mutable Tuple

In Rust, a tuple is immutable, which means we cannot change its elements once it is created. However, we can create a mutable tuple by using the *mut* keyword before assigning it to a variable.

Practice 8: week-8/practice_8/src/main.rs

```rust
fn main() {
    // initialize a mutable tuple
    let mut mountain_heights = ("Everest", 8848, "Fishtail", 6993);

    println!("Original tuple = {:?}", mountain_heights);

    // change 3rd and 4th element of a mutable tuple
    mountain_heights.2 = "Lhotse";
    mountain_heights.3 = 8516;

    println!("Changed tuple = {:?}", mountain_heights);
}
```

Passing a Tuple as parameter

A tuple can be passed by value to functions

Practice 9: week-8/practice_9/src/main.rs

main.rs

```rust
fn main() {

    let b:(i32,bool,f64) = (110,true,10.9);
    print(b);

}

//pass the tuple as a parameter

fn print(x:(i32,bool,f64)) {

    println!("Inside print method");
    println!("{:?}",x);

}
```

Destructuring a Tuple

We can break down tuples into smaller variables in Rust, known as destructuring.

Practice 10: week-8/practice_10/src/main.rs

main.rs

```rust
fn main() {

    let b:(i32,bool,f64) = (30,true,4.9);
    print(b);
}

fn print(x:(i32,bool,f64)) {

    println!("Inside print method");
    //assigns a tuple to distinct variables
    let (age,is_male,cgpa) = x;
    println!("Age is {} , isMale? {},cgpa is {}",age,is_male,cgpa);

}
```

Rust Slice

- A slice is a pointer to a block of memory.
- Slices can be used to access portions of data stored in contiguous memory blocks.
- It can be used with data structures like arrays, vectors and strings.
- Slices use index numbers to access portions of data. The size of a slice is determined at runtime.
- Slices are pointers to the actual data. They are passed by reference to functions, which is also known as borrowing.

Rust Slice

Practice 11: week-8/practice_11/src/main.rs

main.rs

```rust
fn main() {
    // an array of numbers
    let numbers = [1, 2, 3, 4, 5];
    println!("Original array = {:?}", numbers);

    // create a slice of 2nd and 3rd element
    let slice1 = &numbers[1..3];
    println!("2nd and 3rd elements sliced = {:?}", slice1);

    // omit the start index
    let slice2 = &numbers[..3];
   // This means the slice starts from index 0 and goes up to index 3 (exclusive)
    println!("index 0 to index 3 sliceed = {:?}", slice2);

    // omit the end index
    let slice3 = &numbers[2..];
    // This means the slice starts from index 2 and goes up to index 5 (exclusive)
    println!("index 2 to index 5 sliced = {:?}", slice3);

    // omit the start index and the end index
    // reference the whole array
    let slice4 = &numbers[..];
    // This means the slice starts from index 0 and goes up to index 5 (exclusive).
    println!("index 0 to index 5 sliced = {:?}", slice4);
}
```

Mutable Slice in Rust

Practice 12: week-8/practice_12/src/main.rs

main.rs

```rust
fn main() {

    // mutable array
    let mut colors = ["red", "green", "yellow", "white"];

    println!("\nOriginal array = {:?}", colors);

    // mutable slice
    let sliced_colors = &mut colors[1..3];

    println!("First slice = {:?}", sliced_colors);

    // change the value of the original slice at the first index
    sliced_colors[1] = "purple";

    println!("Changed slice = {:?}", sliced_colors);
}
```

Class Projects

SCHOOL OF
SCIENCE AND
TECHNOLOGY
PAN-ATLANTIC UNIVERSITY

Class Project I

You have been invited to join the team of developers to build a Public Service APS level checker for the Federal Government of Nigeria. You have been provided the following table below.

Develop a Rust program using vectors to validate Staff level. E.g. If Staff is an Associate Lawyer, and has 5-8 years of work experience, then the staff holds position APS 5-8.

Public Servant	Office Administrator	Academic	Lawyer	Teacher
APS 1-2	Intern	–	Paralegal	Placement
APS 3-5	Administrator	Research Assistant	Junior Associate	Classroom Teacher
APS 5-8	Senior Administrator	PhD Candidate	Associate	Snr Teacher
EL1 8-10	Office Manager	Post-Doc Researcher	Senior Associate 1-2	Leading Teacher
EL2 10-13	Director	Senior Lecturer	Senior Associate 3-4	Deputy Principal
SES	CEO	Dean	Partner	Principal

Class Project II

Ernst & Young (EY) Global Limited, is a multinational professional services network with headquarters in London and branch offices all around the world. EY is one of the largest professional services networks in the world. EY Nigeria recently launched a project and are scouting for developers with the highest years of experience. You have been contacted by the CEO of EY Nigeria to develop a Rust program to find the person with the highest years of programming experience during the job interview using any rust compound data type.