Lecture 7

Slice Types

Goals For Today

- Slices
- Strings

Array and Slice

- Arrays ([T, N]):
  - Contiguous collection of type T objects
  - Size N known at compile time
  - Live on the stack

- Slices (&[T]):
  - View into (i.e: immutably borrow) any contiguous collection of type T
  - Original collection can live anywhere
  - Size not known at compile time
  - Can borrow a section of an array i.e: subarray

```rust
fn main() {
    // this is an array
    let mut x: [i32; 5] = [1, 2, 3, 4, 5];

    // this is a slice which immutably
    // borrows from x
    let x_slice: &[i32] = &x;

    // this is another slice which immutably
    // borrows x from indices 1 to 3
    let x_slice_section: &[i32] = &x[1..4];

    // ERROR: cannot assign is x because it is borrowed
    x[3] = 6;

    println!("{:?}", x_slice);
}
```

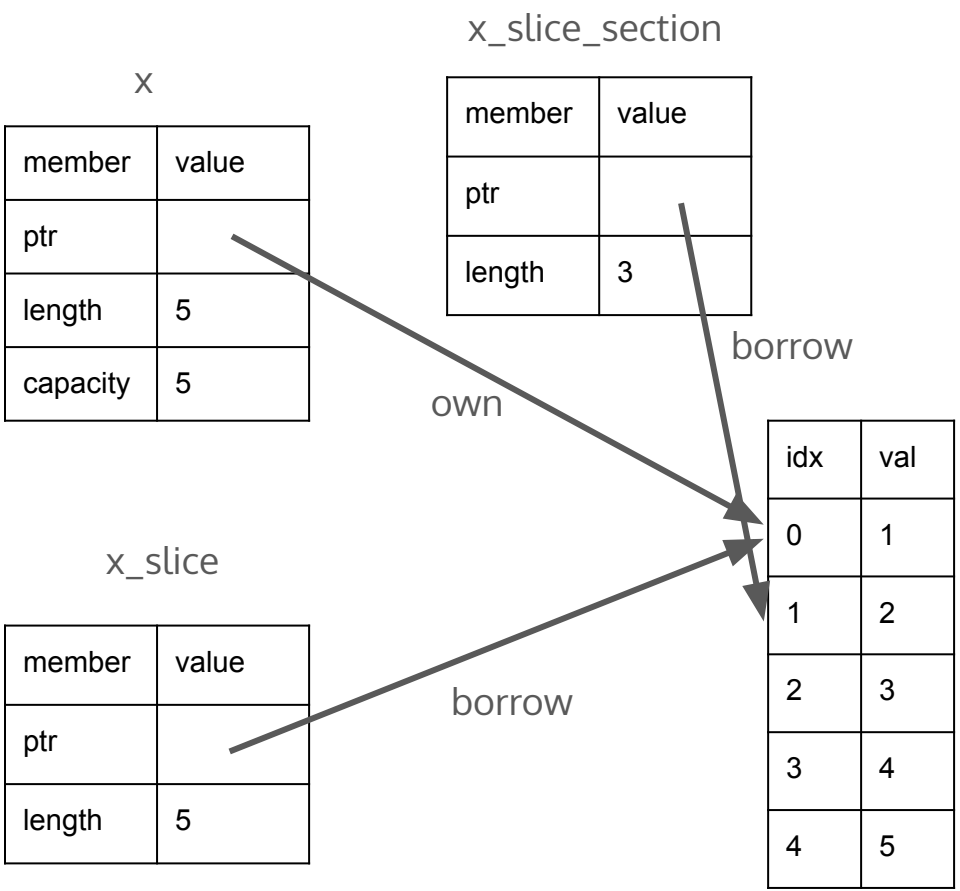
Vec and Slice

- Vec<T>:
  - Mutable, contiguous collection of type T objects
  - Variable size, not known at compile time
  - Stores a heap array

    (size not known at compile time)
- Slices:
  - Can borrow from said heap array
  - Also a borrow of the Vec itself

```
fn main() {
    // this is a Vec
    let mut x: Vec<i32> = vec![1, 2, 3, 4, 5];

    // this is a slice which immutably
    // borrows from x
    let x_slice: &[i32] = &x;

    // this is another slice which immutably
    // borrows x from indices 1 to 3
    let x_slice_section: &[i32] = &x[1..4];

    // ERROR: cannot borrow x as mutable because
    // it is also borrowed as immutable
    x[3] = 6;

    println!("{:?}", x_slice);
}
```

Vec and Slice: Visualized




```rust
fn main() {
    // this is a Vec
    let mut x: Vec<i32> = vec![1, 2, 3, 4, 5];

    // this is a slice which immutably
    // borrows from x
    let x_slice: &[i32] = &x;

    // this is another slice which immutably
    // borrows x from indices 1 to 3
    let x_slice_section: &[i32] = &x[1..4];

    // ERROR: cannot borrow x as mutable because
    // it is also borrowed as immutable
    x[3] = 6;

    println!("{:?}", x_slice);
}
```

Strings and &str: a review

- &str:
  - Immutable

```
let str_example: &str = "Howdy CS 128 Honors";
```

```
let capital_string: String = String::from("Howdy CS 128 Honors");

let ampersand_string: &str = capital_string.as_str();
```

- String:
  - Mutable

```
let howdy_1: String = String::from("Howdy");

let mut howdy_2: String = "Howdy".to_string();

let mut empty: String = String::new();

let five: String = 5.to_string();
```

String and &str revisited

- String:
  - Contiguous collection of UTF-8 characters
  - Variable size, not known at compile time
  - Basically a Vec<u8>
- &str:
  - Also called a “string slice”
  - Basically a &[u8]
  - Can borrow from a String’s heap array of u8’s
  - Also a borrow of the String itself
  - Can be used to borrow substrings
  - Can only be used on Strings (i.e: not Vec<u8>’s)

String and &str: Example

Another way to create &str from String?

```
fn main() {
    // this is a String
    let mut x: String = "Hello CS 128H".to_string();

    // this is a slice which immutably
    // borrows from x
    let x_slice: &str = &x;

    // this is another slice which immutably
    // borrows the substring "ell"
    let x_slice_section: &str = &x[1..4];

    // prints: string 'Hello CS 128H' with substring 'ell'
    println!("string '{}' with substring '{}'", x_slice, x_slice_section);
}
```

Something is missing..

- We have a string analogue to Vec<T>, we have a string analogue to &[T]..
- What about [T, N]?
- We have one, kinda..
- The str primitive type
  - Array-like: contiguous collection of u8's
  - Size not known at compile time
  - Can live anywhere
  - Ex: Static strings and u8 array stored by String
  - Mostly referred to in borrowed/slice form, in fact &str borrows from this type
  - Rarely used otherwise

```
fn main() {
    // we lied, this line creates a static str
    // then assign a slice borrowing this str
    // to `x`
    let mut x: &str = "Hello CS 128H";
}
```

```
fn main() {
    // ascii values for characters: a, b, c
    let x: [u8; 3] = [97, 98, 99];
    // treats x as a stack `str`
    // converts the slice to x into a string slice
    // x_borrow borrows from x
    let x_borrow: &str = str::from_utf8(&x).unwrap();
    // prints: abc
    println!("{}", x_borrow);
}
```

Announcements

HW 5 is released (due 2/26 11:59 PM)

MP 1 is released (due 3/5 11:59 PM)