

Fundamentals | Advanced Memory

■ Intermediate memory refresh

- ◆ All data has a memory address
 - Addresses determine the location of data in memory
- ◆ Offsets can be used to access adjacent addresses
 - Also called indexes/indices

■ Stack

- ◆ Data placed sequentially
- ◆ Limited space
- ◆ All **variables** stored on the stack
 - Not all data
- ◆ Very fast to work with
 - Offsets to access

Stack Visualization



■ Heap

- ◆ Data placed algorithmically
 - Slower than stack
- ◆ Unlimited space (RAM/disk limits apply)
- ◆ Uses pointers
 - Pointers are a fixed size
 - *usize* data type
- ◆ Vectors & HashMaps stored on the heap
 - All dynamically sized collections



10

20

30

40

■ Example

```
struct Entry {  
    id: i32,  
}
```

```
fn main() {  
    let data = Entry { id: 5 };  
    let data_ptr: Box<Entry> = Box::new(data);  
    let data_stack = *data_ptr;  
}
```

Sized Error

```
error[E0746]: return type cannot have an unboxed trait object
--> src/main.rs:1:16
   |
1  | fn sample() -> Fn() {
   |                ^^^^ doesn't have a size known at compile-time
   |
= note: for information on `impl Trait`, see <
```


■ Recap

◆ Stack

- Sequential memory addresses
- Used for variables
- Limited size
- Must know data size ahead of time

◆ Heap

- Algorithmically calculated memory addresses
- Used for large amounts of data
- Unlimited size
- Dynamically sized data/unknown sized data