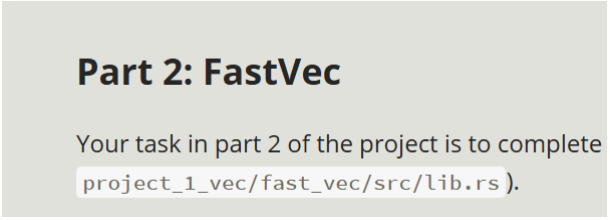
DS 210 Discussion 6 - FastVec

Instruction: https://rust4ds.github.io/ds210-sp26-a1/projects/project01_vec.html

(Scroll down to **FastVec** part.)



1 Part 2: FastVec

Again, check the instructions in details! You will find it very helpful and answer many common issues you may have (use search on webpage).

1.1 Updates

Our code repo again: https://github.com/rust4ds/ds210-sp26-a1-code.

We have updated the **part 2** in the course repo. You will sync it to your fork, **after you merge your `submission1` branch to the `main` branch.**

- There should not be any conflicts and can be merged automatically.

1.2 Works you need to do

1. **Student 1**:
   1. Merge your `submission1` branch to the `main` branch, and **sync the fork from our course repo.**
   2. Implement the `get()` function together.
   3. Create your own branch for implementing `remove()`.
   4. **After both of you have finished,** merge two branches to a `submission2` branch.
   5. Fix the bugs in `clear()` together.
   6. Do the experiment and finish the homework.
   7. Submit the link of github repo to gradescope with **student 2** as collaborator.
2. **Student 2**:
   1. Implement the `get()` function together.
   2. Create your own branch for implementing `push()`.
   3. Fix the bugs in `clear()` together.
   4. Do the experiment and finish the homework.

1.3 Hints

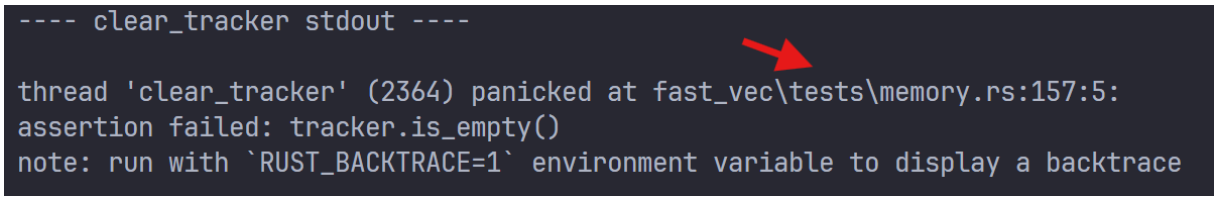
1. In `get()`: You need to return a reference `&T`. Which way should you use to get the value from `ptr_to_data`?
2. In `push()`: You should free the memory of old pointer after `malloc` a new one and copy the old values to the new one. Take a look at `clear()` of how to do that.
3. In `remove()`: Take care of the order of `read()` and `write()`. And don't do double `read()` or `read()` before `write()`.
4. Fix `clear()`: remember how did you deal with the old pointer in `push()`?

1.4 Debugs...

If you get everything correct, **`cargo test -- --test-threads=1`** should give you all **ok**:

```
running 7 tests
test empty ... ok
test get_strings ... ok
test into_vec_numbers ... ok
test into_vec_strings ... ok
test non_empty ... ok
test push_and_read_numbers ... ok
test remove_numbers ... ok

test result: ok. 7 passed; 0 failed; 0

     Running tests\errors.rs (C:\Users\

running 2 tests
test read_out_of_bounds - should panic
test remove_out_of_bounds - should pani

test result: ok. 2 passed; 0 failed; 0

     Running tests\memory.rs (C:\Users\

running 6 tests
test clear_tracker ... ok
test empty_memory ... ok
test push_memory ... ok
test push_tracker ... ok
test remove_memory ... ok
test remove_tracker ... ok

test result: ok. 6 passed; 0 failed; 0

   Doc-tests fast_vec

running 0 tests

test result: ok. 0 passed; 0 failed; 0
```

If you failed in some tests, locate the error in the logs and read the certain lines that throws the error. (You should be able to **Ctrl+click** the code line link the log window to navigate there.)



Also, you can run other tests/bins defined in the `Cargo.toml` (which we also give you in the instructions).

```
[lib]
name = "fast_vec"
path = "src/lib.rs"

[[bin]]
name = "main"
path = "src/main.rs"

[[bin]]
name = "memory"
path = "src/memory.rs"

[[bin]]
name = "experiment"
path = "src/experiment.rs"
```

1.5 Sheet

```
// Allocate
let ptr_to_data: *mut T = MALLOC.malloc(size_of::<T>() * N) as *mut T;

// Read
let val: T = ptr::read(ptr_to_data);
let val: T = ptr_to_data.read()

// Return reference
let val: &T = &*ptr_to_data

// Write
ptr::write(ptr_to_data, val);
ptr_to_data.write(val);

// Move pointer to index of an "array"
let ptr_10: *mut T = ptr_to_data.add(10);

// Free
MALLOC.free(self.ptr_to_data as *mut u8);
```

1.6 Diagram used in class