Tester

Varför testar man kod?

Varför testar man kod?

- Koden är korrekt
- Koden körs tillräckligt snabbt
- Undvika att introducera nya buggar

Olika typer av tester

- Enhetstester/unit tests
- Integrationstester/integration tests
- End-2-end-tester (dörr-till-dörr-tester?)
- Bottom-up

Regression testing

- Man bygger upp en mängd tester under projektets gång
- Förhindrar att buggar återuppstår
- Varje buggfix kan t.ex. ha ett associerat test

Fuzz testing / fuzzing

- Släng slumpmässig input på programmet
- T.ex. kompilatorer/språkverktyg som körs med randomiserad kod
- Exempel: libFuzzer (LLVM)
- Kan analysera vilka delar av koden som påverkas och generera ny input för att täcka fler delar.

Stress testing

- Testar hur en applikation hanterar onormalt höga belastningar
- T.ex. webbservrar

Benchmarking

- Testar applikationens prestanda
- Nödvändigt för realtidssystem (flygplan etc.)
- Kommer vara krav i en av läxorna

Manuella tester

- Om det är för svårt/tidskrävande att konstruera tester för något
- Demo?

Tester i Rust

```rust
fn min_even(nums: Vec<u64>) -> Option<u64> {
  // Return the smallest even number in `nums`,
  // or None if there are no even numbers in `nums`
  todo!("Implement this");
}
```

```rust
#[cfg(test)] // Modulen kompileras endast när tester körs
mod test {
  use super::*; // Importera funktioner etc. från föräldramodulen

  // Tester tester tester...
  #[test]
  fn multiple_even() {
    assert_eq!(min_even(vec![9, 10, 11, 12], Some(10));
  }
}
```

#[should_panic]

```
#[test]
#[should_panic]
fn just_panic() {
  panic!("Aaaaah");
}
```

Automatiska tester

- T.ex. GitHub actions
- Tester körs automatisk för ett pull request
- Om testerna passerar kan PR:en merge:as

Hur har det gått?

- Hur har ni representerat brädet/pjäser etc?
- Hur ser API:t ut? Vilka funktioner finns?
- Hur kan man testa koden?

Läxa

- Fortsätt med schackbiblioteket (inga nya repon)
- **Skriv tester**
- Biblioteket ska vara helt klart till nästa övning

Extra

- Bra video om testing finns här.