VEŽBA 7

Test Driven Development – TDD – je koncept razvoja softvera kod kojeg se testiranje izvodi paralelno sa samim razvojem softvera. Štaviše, testovi se uspostavljaju odmah nakon što je dizajn softvera završen, još pre nego što se funkcionalni deo softvera počne pisati (u teoriji, naravno, u praksi je ovo izmešano u skladu sa realnim potrebama).

Automatizovano izvođenje testova nad relativno malim funkcionalnim celinama programa (engl. *unit*) poznato je i pod nazivom **unit testing**.

Gruba provera uslova u programu

U toku razvoja softvera, obično je pogodno u sam izvorni kod programa uneti provere nekih preduslova koje će prekinuti izvršenje programa ukoliko navedeni uslov nije ispunjen. Pri prekidu programa biće na terminalu ispisano gde je i pod kojim uslovima program prekinu.

Makro funkcija assert je glavni alat pomoću kog se ovo izvodi:

```
#include <assert.h>
...
assert(x > 0); // program će se prekinuti ako nije x>0
```

Dakle, funkcija `assert` prekine izvršavanje programa ako uslov naveden iza makro funkcije `assert` nije ispunjen.

Napomena: makro funkcija `assert` će biti ignorisana ako je definisan makro `NDEBUG`.

Test suite

Da bi se testiranje lakše izvelo, repetitivni deo testova se izdvaja u tzv. test suite koji sadrži funkcije za testiranje i druge pomoćne objekte. Za potrebe ove vežbe, razvijen je jednostavan test suite koji je ovde prisutan u obliku dva fajla sa izvornim kodom: `TestEngine.h` i `TestEngine.c`. Prvi se uključuje u sve module koji obavljaju testiranje, a drugi se linkuje sa test aplikacijom.

Funkcije Test Suite-a

`TEST_START()` – koristi se za započinjanje serije testova.

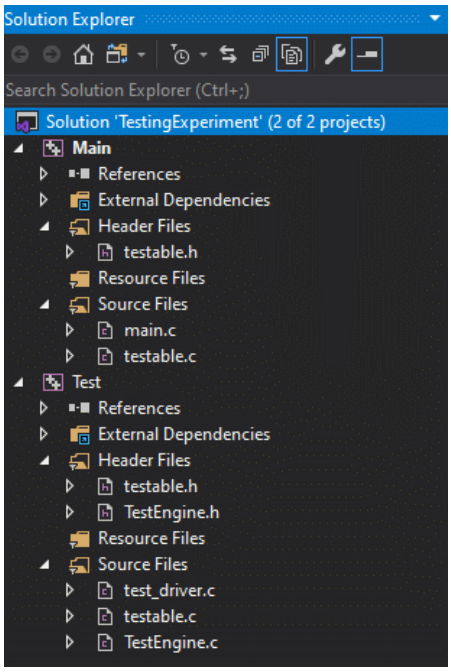
`TEST_END()` – koristi se za završetak serije testova.

`TEST_EQ_INT(a, b)` – sprovodi test jednakosti dva cela broja. Prijavljuje uspešnost ili neuspešnost testa. Ako test nije bio uspešan, prijavljuje detalje oko greške, kao i liniju izvornog koda gde je test pozvan.

`TEST_RANGE_FLOAT(a, b, range)` – sprovodi test jednakosti dva broja u pokretnom zarezu. Pošto brojevi u pokretnom zarezu nisu bezuslovno egzaktni, onda se postavlja opseg unutar kojeg navedeni brojevi treba da se nađu da bi se test smatrao uspešnim (`range`). Prijavljuje uspešnost ili neuspešnost testa. Ako test nije bio uspešan, prijavljuje detalje oko greške, kao i liniju izvornog koda gde je test pozvan.

Postavljanje testa u okruženju Visual Studio

Okruženje Visual Studio upravlja softverskim projektima preko tzv. Solution-a. Jedan solution može da sadrži više projekata, pri čemu jedan izabran kao **startup** projekat koji će se pokrenuti kada se zada komanda Run ili Debug.



Slika 1: Soliution explorer - sa aplikacijom i testom

Ovo će se iskoristiti kada se radi testiranje. Naime jedan projekat će sadržati aplikaciju koja se razvija, a druga će biti aplikacija za testiranje, kako se može videti na slici.

Aplikacija koja se razvija sadržana je u projektu `Main`. Fajl `main.c` je glavni modul koji sadrži funkciju `main` i predstavlja koren aplikacije, dok modul koji sadrži glavnu funkcionalnost programa čine fajlovi `testable.c` i `testable.h` (ovaj drugi opisuje interfejs modula koji omogućava korišćenje njegove funkcionalnosti od strane ostalih modula – ovde glavnog modula).

Aplikacija za testiranje – projekat `Test` – se sastoji iz glavnog modula predstavljenof fajlom `test_driver.c`, test suite-a predstavljenog fajlovima `TestEngine.c` i `TestEngine.h`. U ovu aplikaciju je uključen i modul koji se testira – fajlovi `testable.c` i `testable.h` koji su delovi projekta `Main`.

Osobeni su fajlovi `testable.c` i `testable.h` koji sadrže modul koji se testira. On se fizički nalazi u direktorijumu sa aplikacijom (`Main`), a projekat za testiranje (`Test`) se referencira na iste te fajlove iako su u drugom direktorijumu (zajednički su za oba projekta).

Modul `test_driver.c` treba da ima sledeću strukturu:

```
#include "TestEngine.h"
#include "../main/testable.h"  // referenca na testirani modul

int main()
{
    TEST_START();  // pocetak testiranja
    // niz poziva test funkcija
    // npr. testiranje ispravnosti rezultata funkcije tst_f
    // 5 i 20 su test ulazi, ocekivani izlaz je 10
    TEST_EQ_INT(tst_f(5,20), 10);
    // ... ostali testovi ...
    TEST_END();  // zavrsetak testiranja
    return 0;
}
```

Zadaci

1. Napisati funkciju **`circle_area(float radius)`** koja izračunava površinu kruga na osnovu zadatog poluprečnika. Pomoću **`assert`** skrenuti pažnju ako je poluprečnik zadat kao negativan broj.
2. Napisati test za funkcije **`int fibonacci(int n)`** i **`kvKoren(float x)`** u iterativnoj formi (sa petljom). **`fibonacci`** nalazi zadati n-ti element Fibonačijevog niza i radi sa celim brojevima, pa je rezultat je jednoznačan i egzaktan, a time i jednostavan za testiranje. Za razliku od toga, **`kvKoren`** radi sa tipom `float`, a osim toga rezultat zavisi i od broja iteracija. Shodno tome, testiranje treba da očekuje opseg brojeva sa relativnom greškom od 0,01%. Napisati prazne funkcije tako da testovi ne prođu.
3. Napisati iterativne (sa petljom) verzije funkcije **`fibonacci`** i **`kvKoren`**. Ovaj put, one treba da budu kompletne i ispravne i test konačno treba da prođe kao ispravan.
4. Obe funkcije promeniti u rekurzivnu formu. Iskorititi testove kao pokazatelj da je modifikacije uspešno izvedena.

Pomoć za rešavanje zadataka

Fibonačijev niz

Fibonačijev niz se opisno definiše kao niz čiji je nulti element jednak 0, prvi element jednak 1, a svaki naredni je zbir prethodna dva elementa.

Rekurzivna definicija se može napisati kao:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Heronova metoda izračunavanja kvadratnog korena

Kvadratni koren broja S se može izračunati beskonačnim ponavljanjem sledeće iterativna formule:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{s}{x_n}\right) ,$$

gde je x_{n+1} naredna iteracija proračuna koja se dobija na osnovu rezultata iz prethodne iteracije x_n . Tačna vrednost se dobija kada $n \to \infty$. Početna vrednost x_0 može biti bilo koja pozitivna vrednost. Može se krenuti od vrednosti $x_0 = S$. Postupak se u praksi završava kada razlika između dva uzastopne iteracije zadovoljava $x_n - x_{n+1} \leqslant \varepsilon$ gde je ε tražena apsolutna tačnost izračunatog kvadratnog korena.

Proračun se može izvesti u petlji (iterativno) i rekurzivno.