Modulárny testovač pre súťažné programovanie

Jakub Šimo
Školiteľ: RNDr. Michal Forišek, PhD.

Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského

2019-Jun-26

Príklad úlohy

Rekurzívna fcia

Miško si z Egypta priniesol tri veci: zaručene nelietajúci koberec, výdatnú hnačku, a kúzelnú krabičku, ktorá počíta akúsi rekurzívnu funkciu.

Funkcia je určená nasledujúcim predpisom:

```
if (n>912471559) f(n) = n-912471513; else f(n) = f(f(n+912471514))
```

Vašou úlohou je napísať program, ktorý bude túto funkciu rátať, aspoň pre malé kladné celé čísla.

Input

Vstup má niekoľko, menej ako 5000 riadkov. V každom je jedno kladné celé číslo n, menšie alebo rovné 1 000 000.

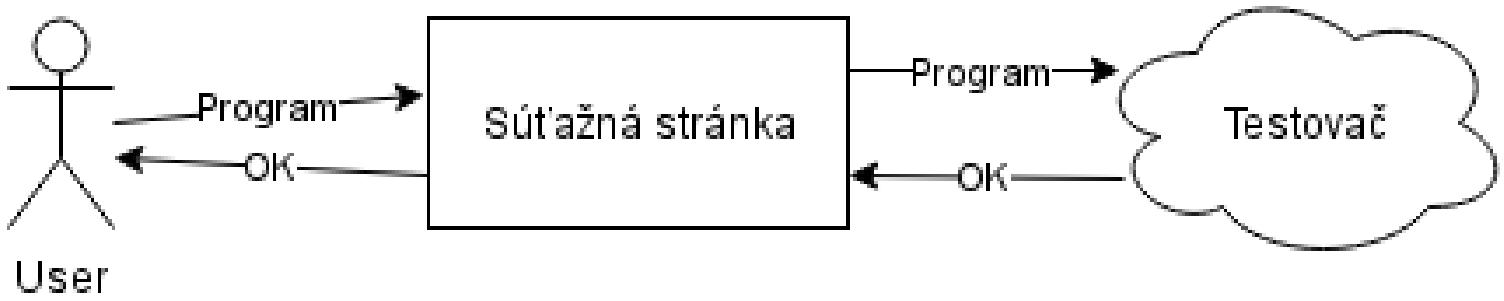
Output

Pre každý vstup vypíšte jeden riadok s príslušnou hodnotou f(n).

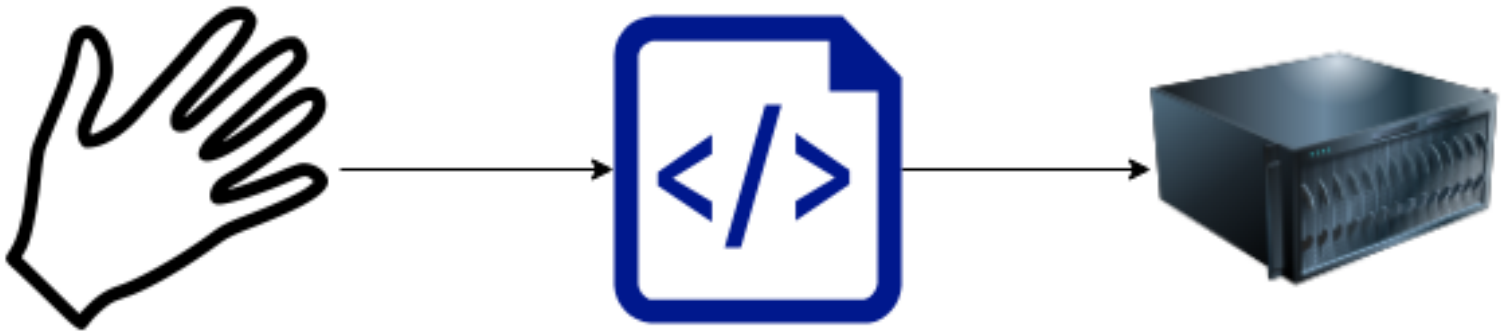
Example

input	output
1234567890	322096377

Princíp problému



Vývoj metodológie testovania



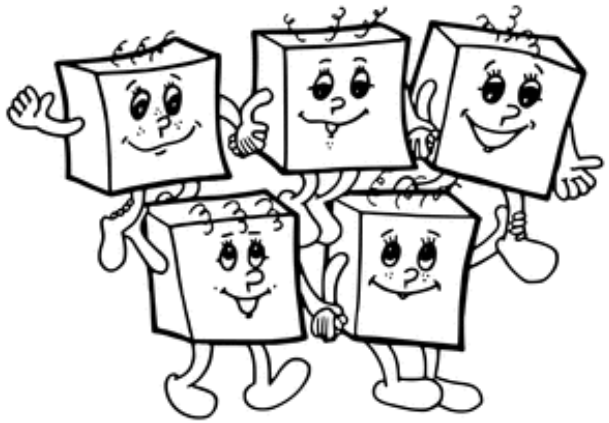
Testovanie

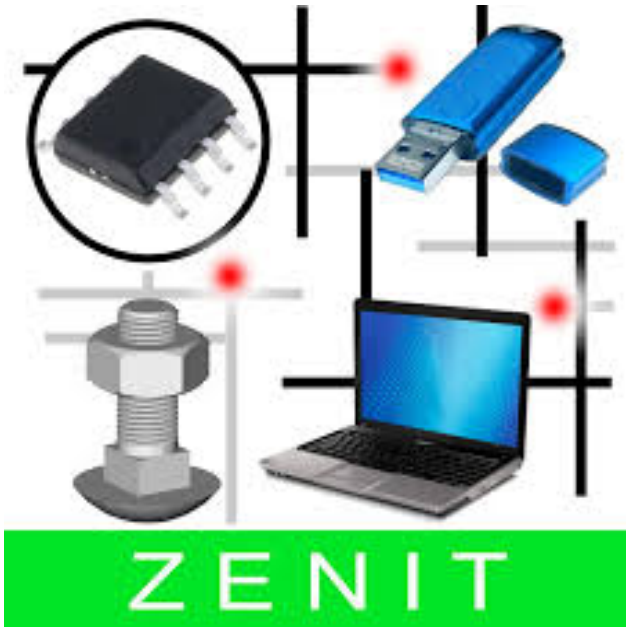
Testovanie

Proces kontrolovania správnosti programu pre určitý problém.

- Dokazovanie správnosti je problém, ktorý sa ešte stále iba skúma
- Pre praktické použitie nám stačí funkcionálne testovanie na sade vstupov

Aktuálne využitia



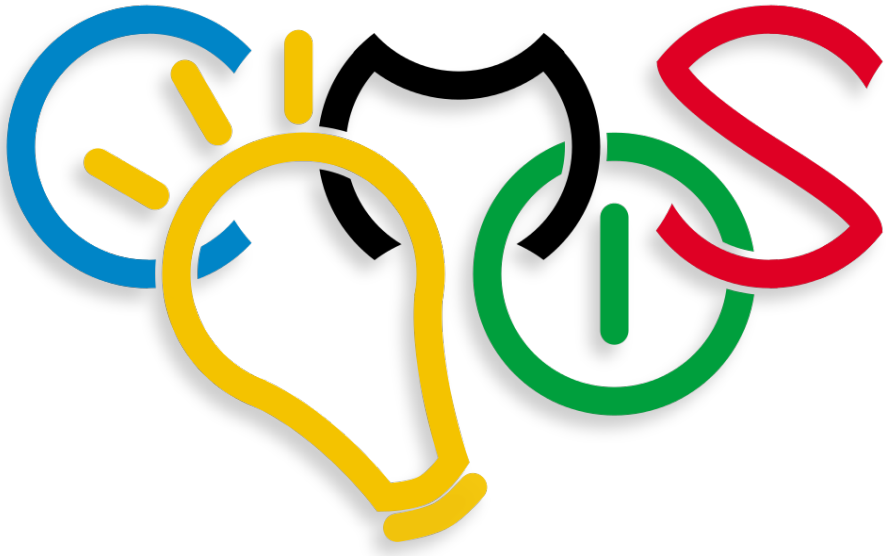


1-INF-810 Rýchlostné programovanie

Aktuálne riešenie

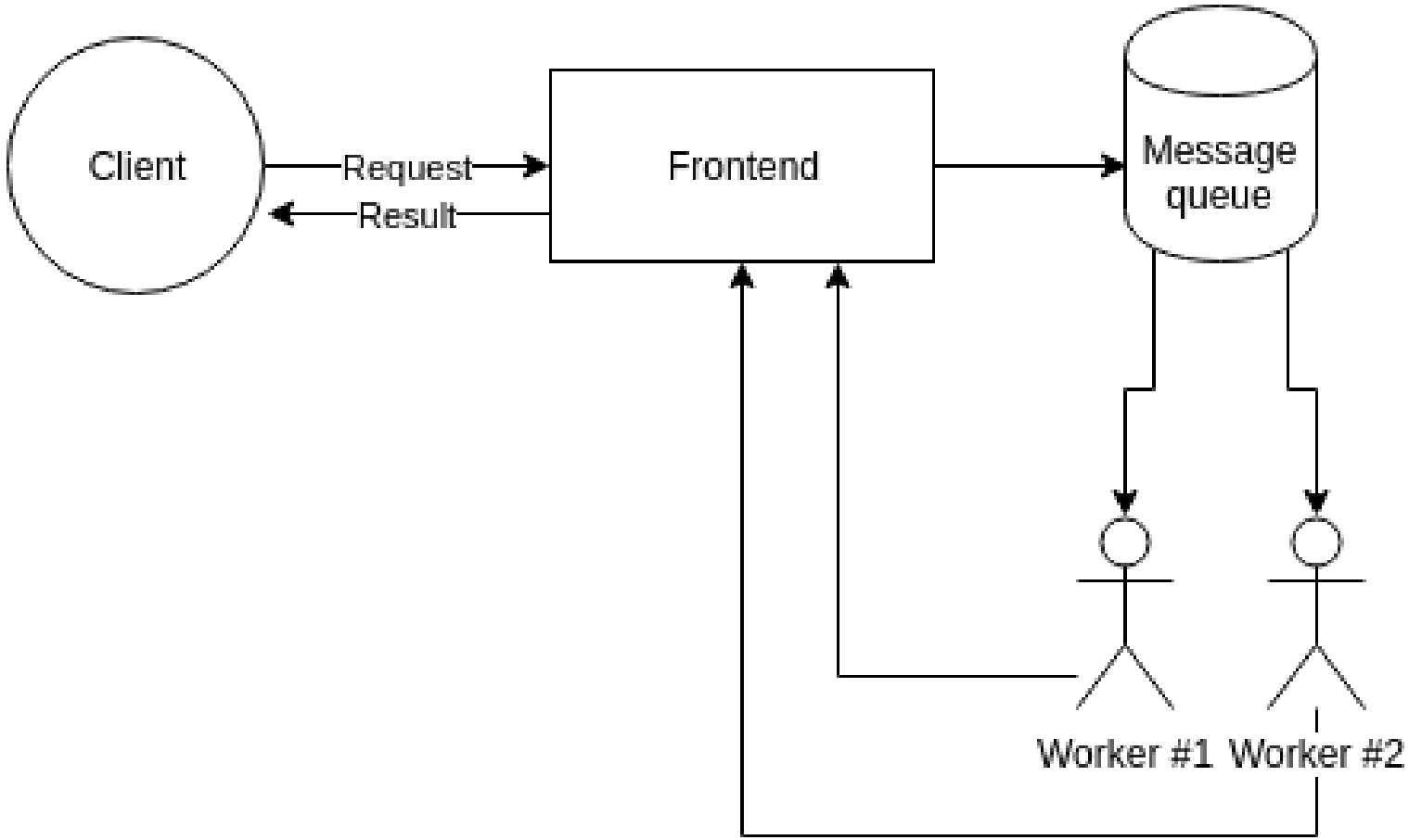
- \+ Prácu vykonáva dobre
- \- Ťažký na rozšírenie kvôli veku, napr. rôzne časy pre jazyky
- \- Nie je možné testovať viacero programov paralelne

Iné riešenia



- celé balíky - stránka, súťaže, testovač
- nemožné použiť pre viacero služieb

Návrh systému



Riešenie viacerých klientov

- rozhranie na žiadanie testovania na strane testovača
  - podobné ako na pôvodnej verzií
- rozhranie na nahlasovanie výsledkov na strane klientov
  - netreba meniť kód testovača pri pridaní nového klienta

Riešenie paralelizácie

- Použitie sprostredkovateľa správ (Message Broker)
- Príjemné vlastnosti ako automatické opakované pokusy s exponenciálnym odstupom pokiaľ je testovací stroj neresponzívny
- Jednoduché pridanie ďalšieho testovacieho stroja - stroj sa prihlási na odber správ od sprostredkovateľa

Riešenie jednoduchšieho pridávania jazykov - Jazykový súbor

```
class Java:
    source_extensions = 'java'
    requires_compilation = True
    compile_command = [
        '/usr/bin/javac',
        'submit.java',
    ]
    run_command = [
        '/usr/bin/java',
        'Submit',
    ]
    memory_overhead = 1912345

    def normalise_time(self, time_limits):
        return time_limits

    def normalise_memory(self, memory_limit):
        return memory_limit + self.memory_overhead
```

Ďakujem za pozornosť

Otázky

1. Bol implementovaný systém testovaný? Ak áno, na akej sade testov a s akým výsledkom?

Zdroj:
https://codegolf.stackexchange.com/questions/69189/build-a-compiler-bomb

```
# 32 TB
(1<<19**8,)*4**7
```

Otázky

2. V práci ste uviedli, že odovzdané riešenie sa bude testovať na všetkých dostupných vstupoch k danej úlohe a až klient sa rozhodne, aký výsledok zobrazí používateľovi. Nie je tento návrh neefektívny, keďže jedným z dôvodov na spájanie vstupov do sád je práve predčasné zastavenie testovania?

Otázky

Inkriminovaná časť textu

"Even if the grader is modular and each client can create their own module, it is unnecessary to do so because each client can be given the full testing report and it can grade the submission according to its needs."