HF01 - PlangTML ellenőrző

Kiíró: Németh Dániel István

Adatszerkezetek és Algoritmusok 2024 ősz

Beadási határidő: 2024 Szeptember 24, 23:59:59

Feladat

A feladat az első két előadáson és gyakorlatokon tanultak felhasználásával egy olyan program megírása, amely ellenőrzi, hogy a bemenetként kapott `string`-ben található PlangTML kifejezés helyes-e.

Egy PlangTML kifejezés nyitó és záró Tagekből áll. Egy nyitó Tag két relációs jel közötti számjegy (`<0>,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>`). A záró tageknél van egy `/` jel is a számjegy előtt (`</0>,</1>,</2>,</3>,</4>,</5>,</6>,</7>,</8>,</9>`). A PlangTML kifejezés helyes, ha a következők mindegyike teljesül:

- Minden nyitó tagnek van záró párja, a nyitó és záró tag számjegyének meg kell egyeznie (pl `<0></0><0>`, `<0></1>` és `<0><1></0>` nem helyesek)
- Maximum egyetlen legkülsőbb tag-pár van (pl `<0></0><1></1>` nem helyes)
- Nincs átfedés különböző tagek által határolt blokkok között (pl `<0><1><3></1></3></0>` nem helyes)

Az alábbi PlangTML kifejezések például helyesek:

- `<4></4>`
- `<0><9></9></0>`
- `<0><1></1><1></1></0>`
- `<0><4><4><5></5></4></4></0>`

Az alábbi szignatúrájú függvényt kell megvalósítani a kiadott skeleton kód `plangtmlchecker.cpp` fájljában:

```
bool PlangTMLChecker::checkCorrectness(
    const std::string& expression
) {
    //TODO solve homework by deadline
}
```

A függvénynek akkor kell `true` értéket visszaadni ha a bemenetként megadott PlanTML kifejezés helyes.

Feltételezhetjük, hogy a függvény bemenetében (`const std::string& expression`) csak nyitó és záró tagek vannak.

A megoldás működésének ellenőrzését néhány teszt is segíti, melyek a *main.cpp* fájlban találhatóak. Ezen teszek eredményei a console-ra íródnak ki a program futtatásakor, a console-ra kiírt pontszám a sikeresen lefutott tesztek számát jelzi, nem a feladatra kapott pontszámot. **FONTOS: A tesztek csak segítik a helyes működés ellenőrzését, de nem fednek le minden esetet, így akár helyesen lefutó tesztek estén is lehet hiba az algoritmusban, megoldásban. Ajánlott írni saját teszteket a működés további ellenőrzésére.**

Ennél a házinál a repo szerveren automata tesztek is futnak, melyek ellenőrzik a skeleton kóddal kiadott teszteket, valamint a memóriaszivárgást is.

A pontozásnál a kódminőség, és az algoritmus hatékonysága is számít a kód helyes működése mellett. Alkalmazzuk a kurzus során tanult módszereket és adatszerkezeteket! A kódba magyarázó kommentek is kerüljenek, amelyek leírják a kód működését! A megoldáshoz STL-t tilos használni. A megadott flagekkel (`-Werror -Wall -Wextra -pedantic`) nem forduló kód esetén a feladat nem értékelhető (0p).