Zh2b-programozás

ⓘ Ez az előnézete a kvíz publikált verziójának

Kezdés: dec 13, 16:27

Kvízinstrukciók

A ZH-n a programozási feladat összesen **30 pont**ot ér, és **120 perc** áll rendelkezésre. Egymással kommunikálni tilos. Segítségként használható a **C referencia** (https://bead.inf.elte.hu/files/c) . A megoldást egy zip fájlba csomagolva kell feltölteni. A zip fájlban csak forrásfájlok legyenek (.c, .h). A feladat megoldása több lépésből áll. Az egyes lépések után kapott megoldást érdemes meghagyni, és a zip fájlba mindegyik elvégzett lépés eredményét becsomagolni.

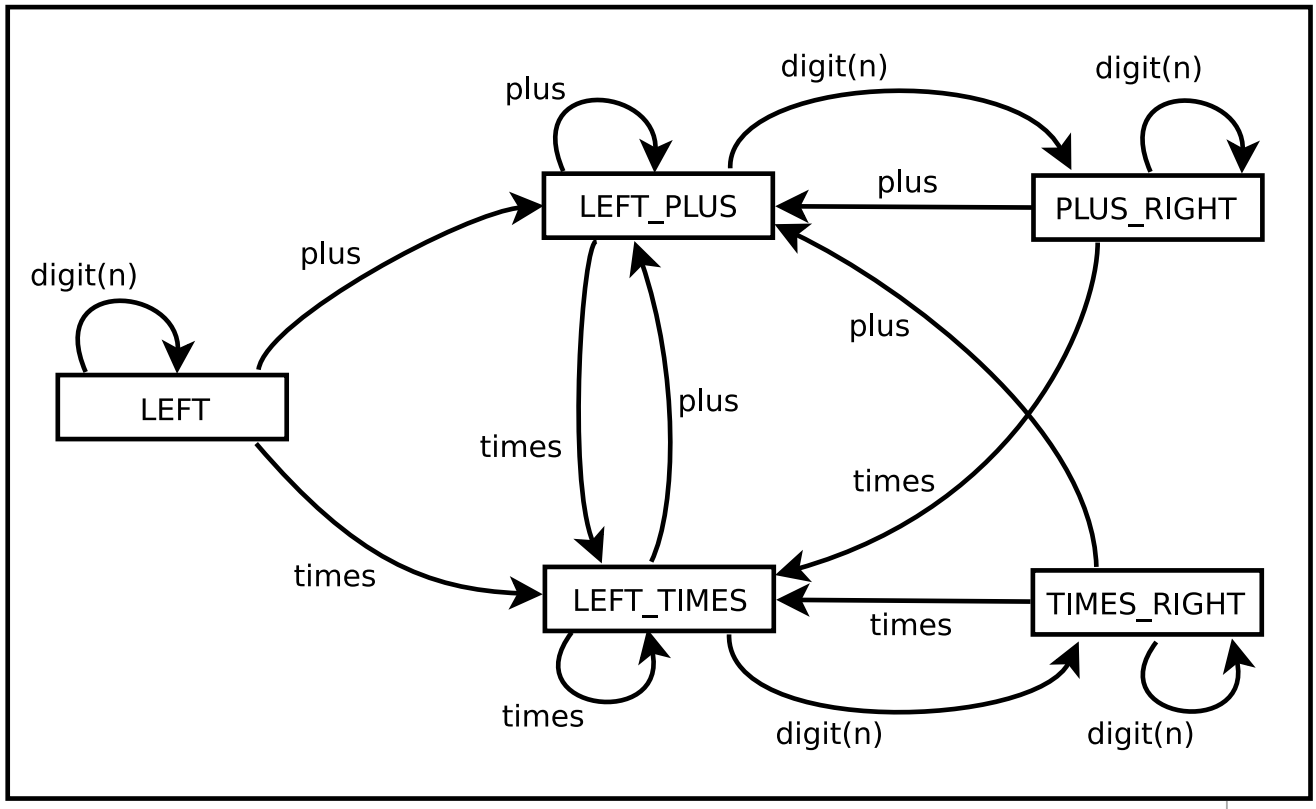
---

1. kérdés — 30 pont

Ki emlékszik még a mobiltelefonok előtt népszerű számológépekre? Készítsünk programot, mely egy ilyen eszközt szimulál! Az egyszerűség kedvéért a számológépünk csak nemnegatív egész számokkal fog dolgozni, és csak az összeadás és szorzás műveleteket támogatja majd. A feladat elsőre kissé bonyolultnak tűnik, de megijedni nem kell. Néhány globális változó és pár egyszerű függvény alkotja a megoldást. A feladatleírás elején elmagyarázzuk, hogyan működik egy számológép, utána pedig részletesen leírjuk, hogy mit is kell csinálni.

Egy számológép működése

A működést az alábbi rajz mutatja be. A gép 5 lehetséges állapottal rendelkezik, melyek között képzeletbeli billentyűk lenyomásának hatására mozog. Számjegyeket (`0`-`9`) és két műveleti jelet (`+` és `*`) vihetünk be a kezelő felületen. A gép legfeljebb két (nemnegatív egész) számot tart a memóriájában: a végrehajtandó művelet két operandusát. Nevezzük ezt a két tárolt számot `memory`-nak és `input`-nak. A számjegyek bevitele (mely a `digit(n)` függvénnyel történik) során az `input`-ban tárolt értéket módosítjuk: az eddigi értéket megszorozzuk 10-zel, és hozzáadjuk a `digit` paraméterét.

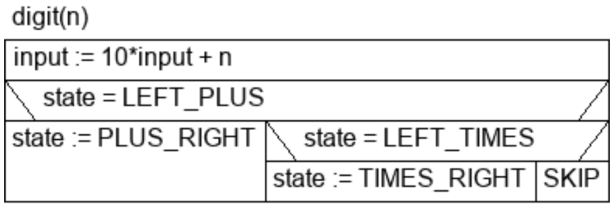


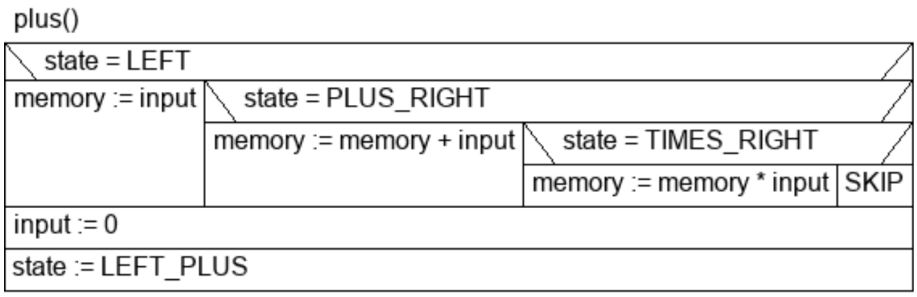
A számológép kiinduló állapota a `LEFT` állapot: ebben az állapotban az első végrehajtandó művelet bal oldali operandusát vihetjük be. Az `input`-ban tárolt érték kezdetben 0, a számjegyek egymás utáni bevitelével hozzuk létre a bal oldali operandust - közben mindvégig a `LEFT` állapotban maradunk. Egy műveleti jel bevitele (mely a `plus` vagy a `times` függvényekkel történik) hatására átlépünk a `LEFT_PLUS` vagy a `LEFT_TIMES` állapotba, és ennek során az `input` értéke átkerül a `memory`-ba, az `input` pedig a 0 értéket veszi fel. Ha egymás után több műveleti jelet viszünk be, a legutoljára bevitt művelet jut érvényre. A `LEFT_PLUS` és a `LEFT_TIMES` állapotokból számjegyek bevitelével juthatunk tovább a `PLUS_RIGHT` és a `TIMES_RIGHT` állapotokba. Ezek a számjegyek az `input` tár módosításával az elvégzendő művelet jobb oldali operandusát határozzák meg. A `PLUS_RIGHT` és a `TIMES_RIGHT` állapotokból egy műveleti jel bevitelével léphetünk tovább, melynek során kiszámoljuk a szóban forgó művelet eredményét, és eltároljuk a `memory`-ban. (Ha a `PLUS_RIGHT` állapotban voltunk, az eredmény `memory+input`, ha a `TIMES_RIGHT` állapotban voltunk, az eredmény `memory*input` lesz.) A `plus` hatására a `LEFT_PLUS`, a `times` hatására a `LEFT_TIMES` állapotba jutunk, és értelemszerűen az `input` 0-ra állításával megkezdhetjük az újabb jobb oldali operandus beolvasását.

Alapfeladat (12 pont)

A fentieknek megfelelően készítsük el a számológépet. Hozzunk létre egy felsorolási típust (`STATE`) az 5 állapot ábrázolására. A számológép reprezentálásához három globális változót használjunk: `state`, `memory` és `input`. (A `memory` és az `input` típusa legyen nemnegatív egész szám.)

Készítsük el a kezelőfelületet megvalósító három függvényt: `digit(n)`, `plus()` és `times()`. A `digit(n)` és a `plus()` struktogramja az alábbi. A `plus()`-hoz használjunk `switch` utasítást!





A `times()` implementációja nagyon hasonló a `plus()`-éhoz, ezért érdemes ezt egy megfelelően felparaméterezett segédfüggvénybe kiemelni. (Segítség: annyi a különbség, hogy a legutolsó utasításban `LEFT_TIMES` értéket kell adni a `state`-nek.)

Készíts egy `reset()` műveletet is, amely alapállapotba hozza a számológépet! (A tárolt értékeket kinullázza, és `LEFT` állapotba lép.)

Készíts főprogramot, mellyel letesztelhetjük a számológép működését! A főprogramban hívjuk meg a kezelőfelület műveleteit egy elképzelt bemenetre, majd írjuk ki a `memory` és az `input` értékét. (Nem kell
fájlból vagy a szabványos bemenetről olvasni az inputot!)

Modularizálás (10 pont)

Bontsuk szét két fordítási egységre a programot! A számológép megvalósítása kerüljön az egyikbe, a főprogram a másikba. A két modul közötti interfész kerüljön a konvencióknak megfelelően egy fejállományba (header). (Használj benne include guardot!)

Az interfészben csak a négy művelet legyen (`digit(n)`, `plus()`, `times()` és `reset()`). A globális változók és az esetleges segédfüggvények legyenek csak a saját fordítási egységükben elérhetők!

A négy művelet adja vissza, hogy mit is látnánk a számológép kijelzőjén! A `reset()` és a `digit(n)` műveletek az `input`, míg a `plus()` és a `times()` műveletek a `memory` értékét adják vissza.

A főprogram a következőképpen működjön. Ha van parancssori argumentuma a programnak, akkor dolgozza fel azokat, mindegyik előtt meghívva a `reset()` műveletet. Egy parancssori argumentum feldolgozása során a benne szereplő karaktereknek megfelelően hívjuk meg a `digit(n)`, a `plus()` és a `times()` műveleteket. (Figyelmen kívül hagyhatók azok a karakterek, amelyek nem a `+`, `*` és `0`-`9`.)

Számológép típus (8 pont)

(Ne töröld ki az eddigi megoldást, készíts új fájl(oka)t ezen részfeladat megoldásához!)

Szabaduljunk meg a globális változóktól, hiszen azok használata kerülendő! Vezessük be a `calculator` típust, mely egy számológép belső állapotát (`state`, `memory` és `input`) egy rekorddal (struktúrával) reprezentálja. A `calculator` legyen egy típusszinoníma a struktúrára! A műveletek kapják meg (be- és kimenő) paraméterként a számológépet!

A főprogramot alakítsd át értelemszerűen!

Feltöltés Fájl kiválasztása

Kvíz mentve ekkor: 16:27 Kvíz beadása