Pótzárthelyi - programozás

Arab szám-római szám átváltó

Oldd meg az alábbi feladatot. Az elkészült .c és .h fájlokat (és csak ezeket a fájlokat) egyetlen **zip** fájlba csomagolva töltsd fel!

Alapfeladat (26 pont)

Készíts el egy programot szabványos C eszközökkel, amely az inputon szereplő arab számokat átváltja római számokká. A program legyen képes 1 és 99 között átváltani a számokat, minden más esetben adjon hibaüzenetet, és lépjen tovább a következő inputra. Azaz, ha az input például:

34

97

67

cica

2

21

0

6

...akkor az output:

XXXIV

XCVII

LXVII

cica: wrong input!

II

XXI

0: wrong input!

VI

A program parancssori paraméterként kapjon egy fájlt, amelyet megnyitva beolvassa a benne szereplő sorokat. Feltételezhetjük, hogy a sorok mindegyike rövidebb 1023 karakternél, és egy sorban csak egy szám (vagy valamilyen hibás input) szerepel. Ha a fájl megnyitása sikertelen, a program írjon ki hibaüzenetet a standard error outputra (*stderr*), majd fejezze be a futást. Hibás sor esetén szintén írjunk ki hibaüzenetet a standard error outputra, majd folytassuk a következő sorral a feldolgozást.

Hozz létre két globális tömböt, amelyek a közül az egyik az egy római számjegyre átváltható arab számokat tartalmazza (1, 5, 10, 50, 100, 500, 1000), a másik pedig a megfelelő római számjegyeket (*I*, *V*, *X*, *L*, *C*, *D*, *M*). Az átváltáshoz használd a globális tömböket: keresd meg az aktuális helyiértéket jelölő római számjegyet (pl. tízes helyiérték esetén 10, a hozzá tartozó római számjegy az *X*), és használd fel a szám átváltásához. Az átváltást helyiértékenként végezd és a legnagyobb helyiértéktől kezdd el. Érdemes az aktuális helyiértéken lévő számjegytől függően több esetre bontani az átváltást. Az egyes helyiérték esetében a kiindulási számjegy az 1, római megfelelője *I*:

* 0: nincs megfelelő számjegy, a függvény térjen vissza NULL-lal vagy üres stringgel.
* 1-3, 5-8: a számjegyek úgy állnak elő, hogy az őket megelőző legnagyobb római számjegyhez megfelelő számú 1-est adunk (I, II, III, V, VI, VII, VIII).
* 4, 9: a számjegyek úgy állnak elő, hogy az utánuk következő számból kivonunk 1-et (IV, IX).

A tízes helyiérték esetében a kiindulási szám a 10, római megfelelője *X*:

* 0: nincs megfelelő számjegy, a függvény térjen vissza NULL-lal vagy üres stringgel.
* 10-30, 50-80: a számok úgy állnak elő, hogy az őket megelőző legnagyobb római számjegyhez (X vagy L) megfelelő számú 10-est adunk (X, XX, XXX, L, LX, LXX, LXXX).
* 40, 90: a számjegyek úgy állnak elő, hogy az utánuk következő kerek számból kivonunk 10-et (XL, XC).

A számokat úgy kezeljük, hogy minden helyiértéket megfelelően váltunk át, tehát pl. a 99 római megfelelője *XCIX* (nem IC).

Az átváltás során keletkező római számot tárold egy megfelelő méretű dinamikus karaktertömbben. A római számokat nem muszáj egy string tömbben eltárolni. A sorok beolvasása legyen hibatűrő, tehát olyan függvényt használj, amely az esetleges rossz inputokat (pl. egy stringet) is be tudja olvasni.

A feladat naiv (brute force) megvalósítására a maximális pontszám legfeljebb fele adható meg.

Modularizálás (4 pont)

Bontsd fordítási egységekre a programot! A main függvény szerepeljen másik fordítási egységben, mint a többi függvény. A main-től különböző függvényeket tartalmazó fordítási egységhez készíts header fájlt is. Ne felejts el header guardot használni!

A globális tömböket helyezd át abba a fordítási egységbe, amelyben az átváltást végző függvények vannak.

Beolvasás több fájlból (5 pont)

A program legyen képes több fájlt feldolgozni parancssori paraméterekként. Ha valamely fájl megnyitása sikertelen, a program írjon ki hibaüzenetet, majd a következő fájllal folytassuk a végrehajtást.

Beolvasás a standard inputról (4 pont)

Ha a programnak nincs parancssori paramétere, a program legyen képes a standard inputról beolvasni sorokat (EOF-ig).

(Tipp: az EOF esemény Unix rendszeren a Ctrl+D, Windowson a Ctrl+Z paranccsal váltható ki.)

Százas helyiértékek átváltása (5 pont)

Egészítsd ki a korábban írt átváltó függvényt úgy, hogy képes legyen 1 és 999 között átváltani az arab számokat. Ha az input:

234

997

567

cica

302

21

0

6

...akkor az output:

CCXXXIV

CMXCVII

DLXVII

cica: wrong input!

CCCII

XXI

0: wrong input!

VI

A százasok átváltását az alapfeladatban szereplő kisebb helyiértékek átváltásával azonos módon kell elvégezni. A különbség, hogy itt a 100 (*C*) lesz az aktuális helyiérték.

A feladat naiv (brute force) megvalósítására a maximális pontszám legfeljebb fele (3 pont) adható meg.

Ezresek átváltása (6 pont)

A program legyen képes 1 és 3999 között átváltani a számokat. Ha az input:

1234

1997

567

cica

302

21

0

6

...akkor az output:

MCCXXXIV

MCMXCVII

DLXVII

cica: wrong input!

CCCII

XXI

0: wrong input!

VI

Az ezres helyiértékeket úgy válthatjuk át, hogy megvizsgáljuk a megadott szám 1000-rel való egészosztásának eredményét, és a kapott számnak megfelelő mennyiségű *M* karaktert írunk a római szám elejére.

Elvárások a programmal szemben

* A nem forduló kód automatikusan 0 pontot ér. (Természetesen ez csak a legutoljára feltöltött megoldásra vonatkozik.)
* Ne használj globális változókat a kért globális tömbökön kívül!
* Logikusan tagold a megoldást. A megoldás részeit külön függvényekben valósítsd meg.
* Ügyelj, hogy ne legyen a programban memóriaszivárgás!
* Kerüld a nem definiált viselkedést okozó utasításokat!

Tanácsok

* Stringek számmá konvertálásához használhatod a long int strtol(const char\* str, char\*\* endptr, int base) függvényt, amely 0-t ad vissza sikertelen konvertálás esetén.
* Ne feledkezzünk meg a dinamikus memóriafoglalás sikerességének ellenőrzéséről, és a foglalt memória felszabadításáról! Teszteld a programot valgrinddal, hogy felderítsd az esetleges memóriaszivárgást.
* Amennyiben a beolvasáshoz az fgets függvényt használod, ne feledd, hogy ha a sor rövidebb, mint a puffer maximális hossza, a függvény beolvassa a sortörést is, ami a kiírásnál hibás eredményhez vezethet.