**Név:** Kis Gergely Domonkos

**Neptunkód**: VMT982

**Email:** [tianarath30@gmail.com](mailto:tianarath30@gmail.com)

6-os csoport

# Feladat:

Készítsen programot és dokumentációt a következő feladathoz:

Két C2L, növekvőleg rendezett listában egy-egy pozitív egész szám prímtényezős felbontása található. Például a 72 esetén a listában szereplő értékek: 2,2,2,3,3. Az egyik szám listájának fejelemére az E1, a másikra az E2 pointer mutat. Készítsen programot, mely E1 listában előállítja a két szám legnagyobb közös osztójának prímtényezős felbontását. E2 listát a művelet közben le kell bontani: vagy át kell fűzni az adott elemet az E1 listába, vagy fel kell szabadítani. Az algoritmus műveletigénye O(n+m) legyen. (A prímtényezős felbontás előállítása is a feladat része

A program C++ nyelven íródjon, a bemeneti adatok fájlban legyenek megadva. Amennyiben lehetséges, használja az előadáson elhangzott műveleteket.

A dokumentáció tartalmazza a feladat megoldásánál használt **algoritmus struktogramját**, a megoldáshoz használt **osztály UML ábráját**, valamint **teszteseteket**.

# Osztályok

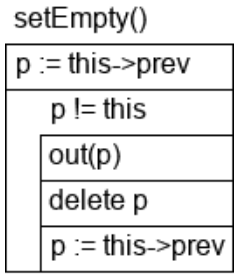
## C2L (C2L.h, C2L.cpp)

|  |
| --- |
| **C2L** |
| +key : ℤ  +prev, next : **C2L\***  +Exceptions : <<***enumeration***>>[FILE\_ERROR,INVALID\_VALUE] |
| +C2L() { prev = next = **this** }  +C2L(key : ℤ) { key = key; prev = next = **this**; }  +~C2L() { setEmpty(); }  +setEmpty()  +print()  +precede(q, r : **C2L\***)  +follow(p, q : **C2L\***)  +out(q : **C2L\***)  +C2L\_read(filename : 𝕊) |

## setEmpty() művelet:

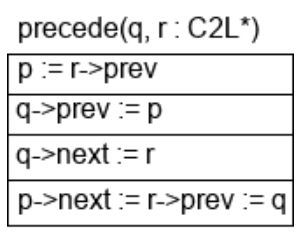
## print() művelet

A **setEmpty()** művelet törli a fejelemen kívüli összes listaelemet.



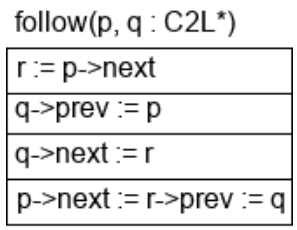
## precede(q, r : C2L\*) művelet

A **print()** művelet a standard kimenetre írja a lista tartalmát



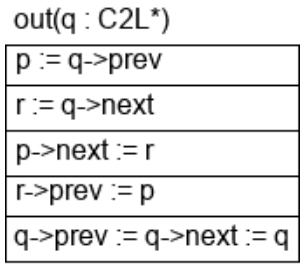
## follow(p, q : C2L\*) művelet

A **precede(q, r : C2L\*)** művelet a lista „r” eleme elé beszúrja a „q” listaelemet



## out(q : C2L\*) művelet

A **follow(p, q : C2L\*)** művelet a lista „p” eleme mögé beszúrja a „q” listaelemet



Az **out(q : C2L\*)** művelet kifűzi a „q” listaelemet a listából

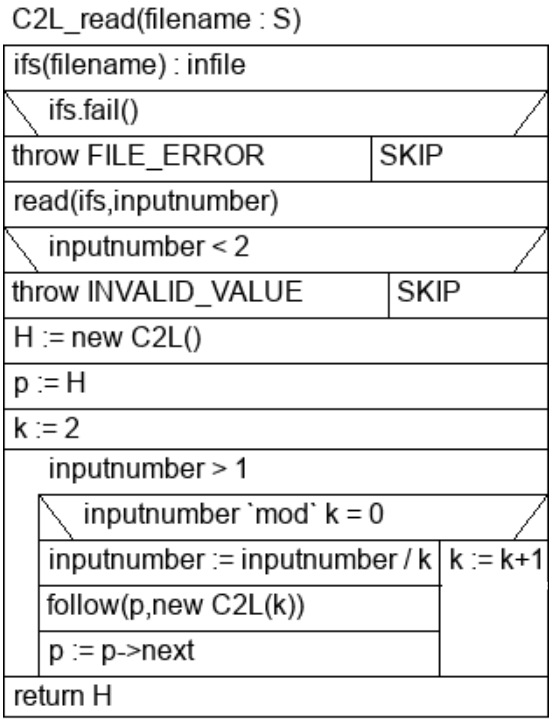
## C2L\_read(filename : 𝕊) művelet

A **C2L\_read(filename : 𝕊)** művelet beolvas egy számot a „filename” paraméterként kapott fájlból és felbontja prímtényezőire, és egy C2L listába teszi őket, amelynek fejelemét végül visszaadja.

A függvény hibával tér vissza amennyiben:

-A kapott fájl nem létezik

-A kapott szám kisebb 2-nél (nincs prímtényezős felbontása)



# Főprogram Algoritmusa

Lnko.h

Lnko.cpp

## largestCommonDivisor(E1: C2L\*, E2: C2L\*)

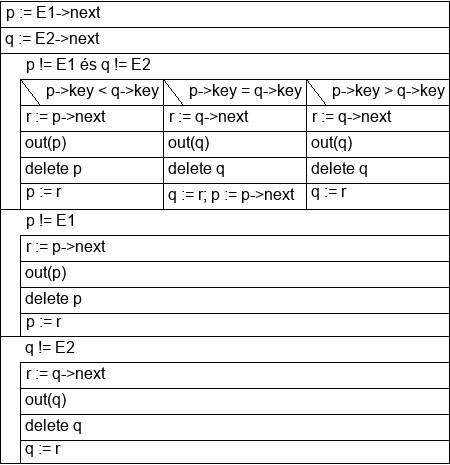
A **largestCommonDivisor(E1: C2L\*, E2:C2L\*)**

alprogram az E1 listában előállítja az E1 és E2 listában tárolt számok legnagyobb közös osztójának prímtényezős felbontását.

A művelet során az E2 listát teljes egészében lebontja.

Amennyiben a legnagyobb közös osztó az 1, akkor az E1 listát is lebontja, és csak a fejelem marad a listában.

Az algoritmus mindkét listát pontosan egyszer járja be így a műveletigény O(N+M)



largestCommonDivisor(E1: C2L\*, E2:C2L\*)

# Tesztelés:

A program tesztelése automatikus tesztkörnyezetben történik. A tesztek kódjai a „tests.h” fájlban találhatóak, a bemeneti tesztfájlok pedig a „tests” nevű mappában.

A tesztelés során a kimenetet a „compareWithResult” nevű függvény összeveti a manuálisan beégetett helyes, várt kimenettel. Csak teljes egyezés esetén megy át a program az adott teszten. A tesztek ellenőrzik továbbá, hogy az E2 lista lebontásra került-e.

A teszteléshez a #define NORMAL\_MODE sort át kell írni   
//#define NORMAL\_MODE-ra, majd így lefordítani a forrásfájlokat.

## Tesztesetek:

### I. Prímtényezős felbontás tesztelése:

1. Nemlétező fájl hiba dobásának tesztelése
2. 2-nél kisebb szám hiba dobásának tesztelése (nincs prímtényezős felbontása)| Test00.txt
3. Páros számok prímtényezős felbontásának tesztelése
   1. 2 - 2 Test01.txt
   2. 72 - 2 2 2 3 3 Test02.txt
   3. 256 - 2 2 2 2 2 2 2 2 Test03.txt
4. Páratlan számok prímtényezős felbontásának tesztelése
   1. 7 - 7 Test04.txt
   2. 105 - 3 5 7 Test05.txt
   3. 231 - 3 7 11 Test06.txt
5. Legnagyobb közös osztó prímtényezős felbontásának tesztelése
   1. eredmények száma szerint:
      1. 0 – Nincs közös prímtényező -> LNKO = 1

E1=15 (3 5) E1=() Test07.txt

E2=8 (2 2 2) E2=() Test08.txt

* + 1. 1 – Egy db közös prímtényező

E1=30 (2 3 5) E1=(2) Test09.txt

E2=22 (2 11) E2=() Test10.txt

* + 1. 2 – Kettő db közös prímtényező

E1=55 (5 11) E1=(5 11) Test11.txt

E2=715 (5 11 13) E2=() Test12.txt

* + 1. Összes (5) db közös prímtényező

E1=4290 (2 3 5 11 13) E1=(2 3 5 11 13) Test13.txt

E2=4290 (2 3 5 11 13) E2=() Test13.txt

* 1. közös prímtényezők elhelyezkedése szerint:
     1. Közösek elöl

E1=231 (3 7 11) E1=(3) Test14.txt

E2=195 (3 5 13) E2=() Test15.txt

* + 1. Közösek középen

E1=770 (2 5 7 11) E1=(5 7) Test16.txt

E2=1365 (3 5 7 13) E2=() Test17.txt

* + 1. Közösek végén

E1=1430 (2 5 11 13) E1=(11 13) Test18.txt

E2=3003 (3 7 11 13) E2=() Test19.txt

* + 1. Közösek szétszórva (+ nagy számok)

E1=510510 (2 3 5 7 11 13 17) E1=(3 7 13 17) Test26.txt

E2=106743 (3 7 13 17 23) E2=() Test27.txt

* 1. prímtényezők multiplicitása szerint
     1. Kettes multiplicitás

E1=12 (2 2 3) E1=(2 2) Test20.txt

E2=20 (2 2 5) E2=() Test21.txt

* + 1. Hármas multiplicitás

E1=135 (3 3 3 5) E1=(3 3 3) Test22.txt

E2=189 (3 3 3 7) E2=() Test23.txt

* + 1. Négyes multiplicitás

E1=4375 (5 5 5 5 7) E1=(5 5 5 5) Test24.txt

E2=6875 (5 5 5 5 11) E2=() Test25.txt