

## DOKUMENTÁCIÓ

**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** :  $\mathbb{Z}$ **+prev, next** : C2L\***+Exceptions** : `<<enumeration>>[FILE_ERROR,INVALID_VALUE]`**+C2L()** { prev = next = **this** }**+C2L(key :  $\mathbb{Z}$ )** { key = key; prev = next = **this**; }**+~C2L()** { setEmpty(); }**+setEmpty()****+print()****+precede(q, r : C2L\*)****+follow(p, q : C2L\*)****+out(q : C2L\*)****+C2L\_read(filename : S)**

## DOKUMENTÁCIÓ

## setEmpty() művelet:

setEmpty()
p := this->prev
p != this
out(p)
delete p
p := this->prev

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

## print() művelet

print()
p := this->next
p != this
write(p->key + " ")
p := this->next

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

## precede(q, r : C2L\*) művelet

precede(q, r : C2L*)
p := r->prev
q->prev := p
q->next := r
p->next := r->prev := q

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

## follow(p, q : C2L\*) művelet

follow(p, q : C2L*)
r := p->next
q->prev := p
q->next := r
p->next := r->prev := q

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

## out(q : C2L\*) művelet

out(q : C2L*)
p := q->prev
r := q->next
p->next := r
r->prev := p
q->prev := q->next := q

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

## DOKUMENTÁCIÓ

## C2L\_read(filename : S) művelet

C2L_read(filename : S)	
ifs(filename) : infile	
ifs.fail()	
throw FILE_ERROR	SKIP
read(ifs,inputnumber)	
inputnumber < 2	
throw INVALID_VALUE	SKIP
H := new C2L()	
p := H	
k := 2	
inputnumber > 1	
inputnumber mod k = 0	
inputnumber := inputnumber / k	k := k+1
follow(p,new C2L(k))	
p := p->next	
return H	

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 prímtényezőzős felbontása)

## Főprogram Algoritmus

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

largestCommonDivisor(E1: C2L*, E2:C2L*)		
p := E1->next		
q := E2->next		
p != E1 és q != E2		
p->key < q->key	p->key = q->key	p->key > q->key
r := p->next	r := q->next	r := q->next
out(p)	out(q)	out(q)
delete p	delete q	delete q
p := r	q := r; p := p->next	q := r
p != E1		
r := p->next		
out(p)		
delete p		
p := r		
q != E2		
r := q->next		
out(q)		
delete q		
q := r		

Lnko.h

Lnko.cpp

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

alprogram az E1 listában előállítja az E1 és E2 listában tárolt számok legnagyobb közös osztójának prímtényezőzős 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  $O(N+M)$

**DOKUMENTÁCIÓ****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
  - a. 2 - 2 Test01.txt
  - b. 72 - 2 2 2 3 3 Test02.txt
  - c. 256 - 2 2 2 2 2 2 2 2 Test03.txt
4. Páratlan számok prímtényezős felbontásának tesztelése
  - a. 7 - 7 Test04.txt
  - b. 105 - 3 5 7 Test05.txt
  - c. 231 - 3 7 11 Test06.txt
5. Legnagyobb közös osztó prímtényezős felbontásának tesztelése
  - a. eredmények száma szerint:

- i. 0 – Nincs közös prímtényező -> LNKO = 1

E1=15 (3 5)	→	E1=()	Test07.txt
E2=8 (2 2 2)		E2=()	Test08.txt

- ii. 1 – Egy db közös prímtényező

E1=30 (2 3 5)	→	E1=(2)	Test09.txt
E2=22 (2 11)		E2=()	Test10.txt

- iii. 2 – Kettő db közös prímtényező

E1=55 (5 11)	→	E1=(5 11)	Test11.txt
E2=715 (5 11 13)		E2=()	Test12.txt

- iv. 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

**DOKUMENTÁCIÓ**b. közös prímtényezők elhelyezkedése szerint:

## i. Közös elöl

E1=231 (3 7 11)	→	E1=(3)	Test14.txt
E2=195 (3 5 13)		E2=()	Test15.txt

## ii. Közös középen

E1=770 (2 5 7 11)	→	E1=(5 7)	Test16.txt
E2=1365 (3 5 7 13)		E2=()	Test17.txt

## iii. Közös végén

E1=1430 (2 5 11 13)	→	E1=(11 13)	Test18.txt
E2=3003 (3 7 11 13)		E2=()	Test19.txt

## iv. Közös 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

c. prímtényezők multiplicitása szerint

## i. Kettes multiplicitás

E1=12 (2 2 3)	→	E1=(2 2)	Test20.txt
E2=20 (2 2 5)		E2=()	Test21.txt

## ii. Hármass multiplicitás

E1=135 (3 3 3 5)	→	E1=(3 3 3)	Test22.txt
E2=189 (3 3 3 7)		E2=()	Test23.txt

## iii. 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