Dokumentáció

LNBZVX – 1. Baedandó

Készítette: Balogh Ferenc

1. Feladat

Adott egy egész számokat tartalmazó négyzetes mátrix. Állapítsuk meg van-e a mátrixnak olyan oszlopa, ahol a főátló alatti elemek mind nullák. A bemeneten az első adat a mátrix mérete, majd ezt követően sorfolytonosan következnek az értékek.
(Feladatsor 9. feladata)

2. Informális specifikáció

A közvetkezőkben pontokba gyűjtöm azokat a gondolatokat, melyek alapján megterveztem és megvalósítottam a programot.

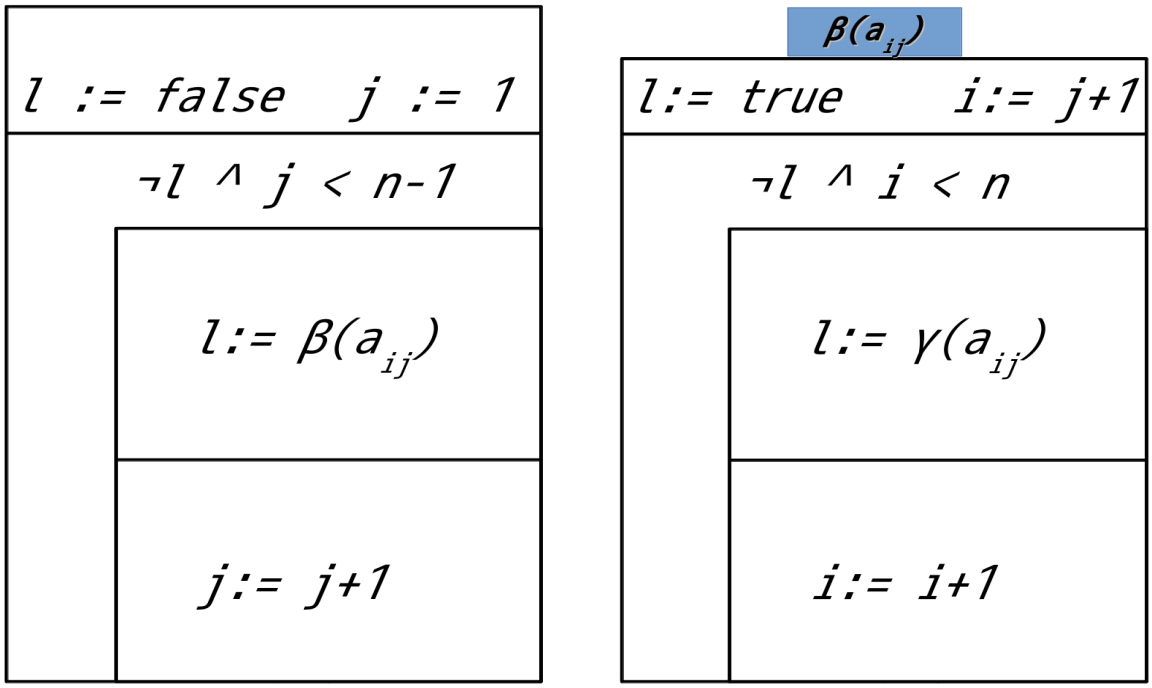
- A beolvasandó adatokat célszerű egy $n \times n$ -es mátrixban eltárolni. Mivel nagyon sokszor volt dolgom azonos típusú elemek táblázatos eltárolásával, célszerűnek láttam egy újrahasznosítható *struct*-ot megírni, amit kiegészítettem egy általános *print_t()* kiíró függvénnyel. Két okból választottam ezt a megoldást: nem kívántam előre szaladni az OOP területre és egyébként is ajánlottnak éreztem a *struct*-tal való közelebbi megismerkedést. Az „általánosságot” *template*-tel valósítottam meg, így a *string*-et leszámítva minden beépített típust kezel a táblázatom. Mivel ez nem volt követelmény a beadandó során ezért a továbbiakban már nem részletezem mélyebben, annyit viszont szeretnék megjegyezni, hogy már ezalatt a beadandó alatt kifejezetten gyümölcsöző volt az, hogy ez a táblázat típus az általános kiíró függvénnyel alapból a rendelkezésemre állt.
- A beolvasás három különböző módon került megvalósításra: *argumentum*-ból, *std-bemenet*-ről illetve *file*-ból való beolvasásból. Arról nem vagyok meggyőződve, hogy a követelmény megkívánja az első megoldást, de mivel végül is elkészítettem benne is hagytam. A *std-bemenet*-ről történő beolvasás nagyon is *nem sorfolytonos* szemben a feladat követelményének, amit számos érv indokol: a beolvasás rendezettebb áttekinthetőbb és éppen ezért sokkal hibatűrőbb, például azt is észre veszi hol romlik el esetlegesen a beolvasás, így lehetőséget teremt az azonnali kijavításra. A *file-beolvasás*-nál a forma viszont lehet akár sorfolytonos vagy akár rendezett mátrix alakban is. Ebből kifolyólag sokkal kisebb a hibatűrése is.
- A feltétel vizsgálatát két egymásba ágyazott lineáris kereséssel valósítottam meg, melyhez a *lineáris keresés* programozási tételt használtam fel. Ezek voltak a sor illetve az oszlop szerinti keresés, melyeket az implementáció során egyetlen függvénnyel valósítottam meg. Ez érthető figyelembe véve azt, hogy a mátrix egy egységes önálló struktúrát alkot.
- Végezetül éltem egy olyan lehetőséggel, amit a feladat egyszerűsége kínált a számomra. Ha a keresés találathoz ér a program ezt jelzi és befejezi futását, hiszen egyéb feladata már nincsen. Olyan programnál, ahol a futás itt nem ér véget extra feltételekkel és elágazásokkal egy un. „egérúton” ki kellene vezetni a programot a keresésből. Mivel ennél a programnál ilyen esetben már nincs egyéb tenni való, ez csak felesleges plusz sorokat jelentene.

3. Specifikáció

A feladat törzsét az a vizsgálat alkotja, hogy „van-e a fődiagonális alatti csupa nulla elemből álló

oszlop”. Ezt egy *lin_ker()* nevű függvényben valósítottam meg, aminek a specifikációját itt tárgyalom.
A megoldás során a *lineáris keresés* programozási tételt használtam fel, kétszer mégpedig egymásba ágyazva: egyszer az oszlop szerinti keresésnél, majd az oszlop elemein végzett keresésnél. Két logikai függvényt használtam, az oszlopokon értelmezett β és az oszlopok elemein értelmezett γ függvények. A második beágyazott stuktogram gyakorlatilag a β működését írja le.

$\beta:\mathbb{N}\to L \;;\; m_{ij}:i,j\in[1,\dots n]\wedge a_{ij}\in\mathbb{Z} \;;\; l\in L$ $\gamma:\mathbb{N}\to L \;;\; m_{ij}:i,j\in[1,\dots n]\wedge a_{ij}\in\mathbb{Z} \;;\; l\in L$



4. Implementáció

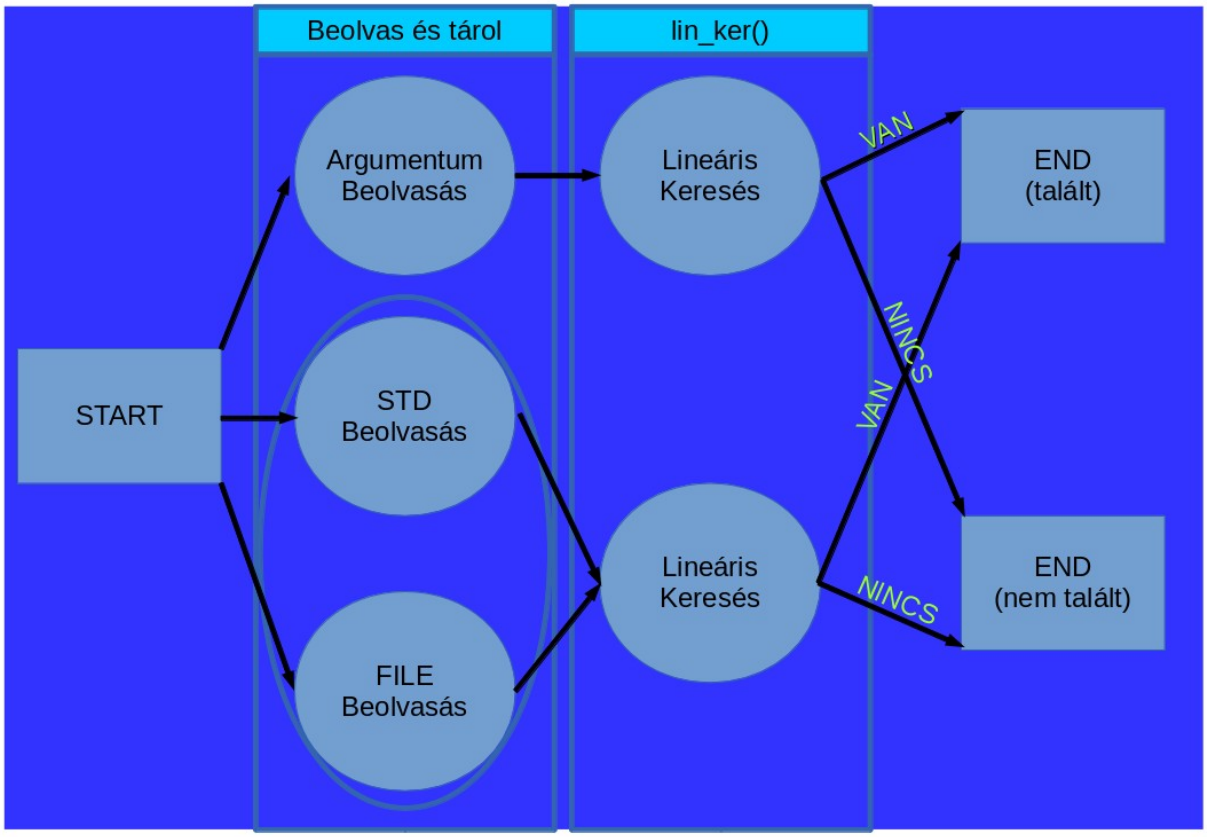
A program két fileból áll:

- prog.cpp : tartalmazza a main() -t és a lin_ker() függvényt
- ss_table.h : tartalmazza a struct table -t, a hozzá tartozó pfint_t() kiíró függvényt, továbbá még a két beadandó specifikus read_std() és read_file() beolvasó függvényeket

Lényegében a fent említett struktúra illetve a függvények szisztematikus hívása alkotja magát a programot, amit még kiegészítettem néhány elemmel bent a *main()* -ben. Röviden ismertetem ezeket a kiegészítő elemeket és a függvényhívások menetét. Teljesebb képet kaphatunk az alább megtekinthető ábrát áttanulmányozva.

A *main()* négy részre tagolható: a program indulása (alapadatok és információk), *if-else* elágazás egy-egy része, és végül a negyedik befejező rész, amihez abban az esetben érkezik el a program, ha a keresés nem hozott találatot.
Az elágazás *if*-része az argumentum beolvasásnál aktiválódik, ha az argumentumok száma nagy. Ha a beolvasás sikeres az adatokat eltárolja a *table<int> M(w,w)* négyzetes mátrixba, ahol *w* a mátrix mérete, majd ezután végrehajtja rajta a keresést, a lin_ker() függvény hívással.
Az elágazás *else*-része az aktiválódása után a beolvasáshoz felkínálja a választást az *std-bemenet* és a *file-bolvasás* között. Ezután megtörténik a beolvasás és mindkét esetben ugyan ahhoz a *lin_ker()* függvényhíváshoz lyukadunk ki.

A teljesség jegyében a 2. pont utolsó bekezdését itt is tárgyalom. A *lin_ker()* függvény zárja mindegyik futási szálat és mikor meghívásra kerül két lehetőség áll fenn. Sikeres találat esetén önmaga befejezi a program futását és jelzi a felhasználó felé a sikert. Ellenkező esetben tovább engedi a program futását amivel sor kerül a befejező részre. Ha a befejező rész aktiválódik az annyit jelent, hogy a keresésnek nem volt pozitív eredménye, amiről tájékoztatja a felhasználót és lezárja a programot.



5. Tesztelés

A, Argumentum beolvasás (sikeres):

$ ./lnbzvx_1_bead.out 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
...
>A teljes matrixot atvizsgaltam, de nem talaltam a feltetelnek megfelelo oszlopot.
>A program ezzel befejezte futasat es kilep.

B, Argumentum beolvasás(sikertelen):

$ ./lnbzvx_1_bead.out 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
>Nagy szamu parametert kaptam az argumentumban. Feltetelezem, hogy ezek adatok es megprobalom feldolgozni.
>A bemenet nem megfelelo! Kilepes!

C, STD-beolvasás (sikeres):

```
$ ./lnbzvx_1_bead.out                   //meret 3; opcio std (azaz a std-beolvasás utasítása)
                        //Mátrix:
                        //1 2 3
                        //0 1 2
                        //0 -1 0
>Talaltunk az also haromszogben csupa nulla oszlopot, megpedig az elso ilyen: 1-edik oszlop.
>A keresest ezzel befejeztuk és a program sikeresen lefutott.
```

D, STD-beolvasás (sikertelen és hibás bevitel):

```
$ ./lnbzvx_1_bead.out                   //meret 3; opcio std (azaz a std-beolvasás utasítása)
                        //Mátrix:
                        //1 2 3 4
                        //s 2 3
>A 1-ik elem nem megfelelo! Kerem irja be ujra ezt a sort, de csak a 1-ik elemtol kezdve.
                        //1 2 3
                        //1 2 3
>A tablazat tartalma:
>1 2 3
>1 2 3
>1 2 3
...
>A teljes matrixot atvizsgaltam, de nem talaltam a feltetelnek megfelelo oszlopot.
>A program ezzel befejezte futasat es kilep.
```

E, FILE-beolvasás (hibás filenév és sikeres):

```
$ ./lnbzvx_1_bead.out                   //meret 3; opcio a (azaz egy tetszőleges karakter)
                        //megadott filenév: a (azaz 'a' betű kiterjesztés nélkül)
>A file nem letezik vagy rossz nevet adott meg.
>Kerem adjon egy letezo filenevet:
                        //a.txt
...
>Talaltunk az also haromszogben csupa nulla oszlopot, megpedig az elso ilyen: 2-edik oszlop.
>A keresest ezzel befejeztuk és a program sikeresen lefutott.

/*ahol is a file tartalma:
 *1 2 3
 *0 9 4
 *1 0 8
*/
```

Az összes hibalehetőség tesztelése, nagyon sok munka és annál is több dokumentálandó kimenet, ami meghaladja ennek a dokumentációnak a kereteit. A teljes hibatűrés viszont messze nagyobb mint amit ez a tesztsorozat bemutat.

6. Környezet és egyebek

Végezetül néhány rövid mondatban szeretném összefoglalni a háttérkörnyezetet illetve a beadandó

tartalmát.
Az operációs rendszer, amelyen a beadandó készült FEDORA19, telepített 'c++' fordítóval ami a C++11 fordítóval dolgozott. A forráskódot VIM szerkesztővel, a dokumentációt pedig LibreOffice-szal készítettem el.
A beadandóba bekerült file-ok illetve azok tartalma a következők:

- dokumentacio.pdf : azaz ez a dokumentáció
- lnbzvx_1_bead.out : az általam fordított program
- prog.cpp : a main() -t tartalmazó forráskód
- ss_table.h : a 'struct table' -t és a hozzá tartozó függvényeket tartalmazza
- ss_table.h.gch
- a.txt : tesztfile a FILE-beolvasáshoz
- source-prog.pdf : forráskód PDF-ben
- source-ss_table.pdf : forráskód PDF-ben