# Feladat

Egy általános iskola alsó tagozatán papírgyűjtő versenyt rendeztek. A verseny 2018. szeptember 1-től, december 31-ig tartott. Feljegyezték a verseny adatait, és egy szöveges állományban tárolták el. A fájl egy sorának felépítése: elsőként a tanuló neve szerepel (két vagy több szóközök nélküli sztring), majd az osztálynak az azonosítója (1-4 számjeggyel kezdődő, szóközt nem tartalmazó sztring, például 1a, 2b, 4c), majd a papírgyűjtés adatai: dátum-súly (a dátum EEEE/HH/NN alakú sztring, a súly egy pozitív valós szám: a gyűjtött papír súlya kilogrammban megadva) formájában. A sor dátum szerint rendezett. Az adatok szóközökkel vagy tabulátorjelekkel vannak egy soron belül elválasztva. A szöveges állomány sorait osztály-azonosító szerint rendezték. Feltehetjük, hogy a szöveges állomány helyesen van kitöltve. Példa az állomány egy sorára:

Nagyon Szorgalmas Eszter 4c 2018/09/10 4.5 2018/09/22 3.5 2018/11/05 1.2

(1) Listázzuk ki azokat a tanulókat (nevüket és osztályukat), akik legalább egyszer, több mint 5 kg papírt hoztak!

(2) Hány olyan osztály van, melynek minden tanulója (aki a versenyben részt vett) legalább egyszer, több mint 5 kg papírt hozott?

# (1) Részfeladat megoldása

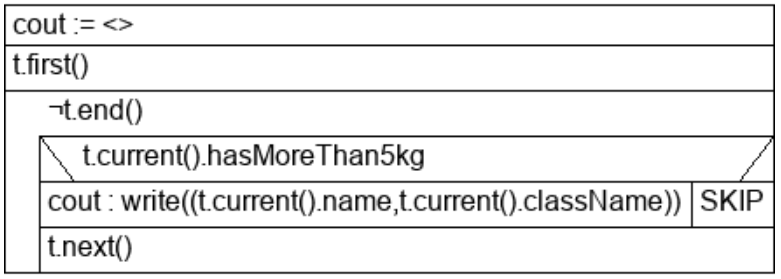
## Főprogram terve

*A* = ( t : enor(StudentPaperData), cout : outfile(StudentPaperDataOut) )

StudentPaperData = **rec**(name : 𝕊, className : 𝕊, hasMoreThan5kg : 𝕃 )

StudentPaperDataOut = **rec**(name : 𝕊, className : 𝕊)

*Ef* = ( t = t’ )

*Uf =* ( cout 

Összegzés (kiválogatás)

## Diákok papírgyűjtési adatainak felsorolója

Felsoroló:

t:enor(StudentPaperData) Contribution = **rec**(name: 𝕊,className: 𝕊,data: 𝕊)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentPaperData\* | first() | next() | current() : StudentPaperData | end() : 𝕃 |
| x : infile(Contribution) dx : Contribution sx : Status act : StudentPaperData end : 𝕃 | sx,dx,x:read  next() | lásd külön | **return** act | **return** end |

StudentPaperData = **rec**(name: 𝕊,className: 𝕊,hasMoreThan5kg: 𝕃)

Az enor(StudentPaperData) first() művelete beolvas egy sort a read() művelet segítségével, majd meghívja a next() műveletet.

A read() művelet az „x” szekvenciális inputfájl egy sorát beolvassa, és az „sx” státuszváltozót állítja „norm” értékre sikeres olvasás esetén, „abnorm” értékre sikertelen olvasás esetén.

Először beolvassa az egész sort egy változóba („iss”) , majd azt kezdi el feldolgozni. Egy másik változóba kezdi el gyűjteni a beolvasott nevet, mivel több tagból is állhat („oss”). Az „iss” változó értékét szavanként összegzi az „oss” változóba, ameddig a beolvasott érték első karaktere nem egy 1 és 4 között lévő szám, azaz a tanuló osztályának azonosítója. Ezt követően a „dx” változó „name” mezőjének a képzett összeget adja („oss”-teljes név), a „className” mezőjének pedig az utoljára beolvasott osztályazonosítót. A „data” mező a hátralévő karakterlánc értékét kapja meg. Amennyiben bármelyik lépés sikertelen a „sx” változó „abnorm” értéket kap.

A next() művelet először beállítja az „end” változó értékét az „sx” státuszváltozó értékétől függően, igaz értékre amennyiben „abnorm” , és hamis értékre amennyiben „norm”. Ezt követően beállítja a felsorolás aktuális elemének („act”) „name” és „className” mezejét a beolvasott „dx” változó „name” és „className” mezőjének értékeire. Utána a „dx” változó „data” mezőjét szavanként (szóközökkel elválasztva) elkezdi feldolgozni a következőféleképpen:

Egyszerre két szót dolgoz fel: az első szó a hozott papír dátuma, a második pedig a hozott papír súlya. Ezeken a párokon végez pesszimista lineáris keresést (eldöntés) úgy, hogy amennyiben talál egy olyan párt, amelyhez tartozó súly nagyobb, mint 5 kg akkor a felsorolás aktuális elemének („act”) „hasMoreThan5kg” mezejét igazra állítja, különben pedig hamison hagyja. A feldolgozást követően végez mégegy olvasást „read()”, ezzel léptetve a felsorolást.

Next Művelet:

*Anext* = (x:infile(Contribution),dx:Contribution,sx:Status,act:StudentPaperData,end: 𝕃)

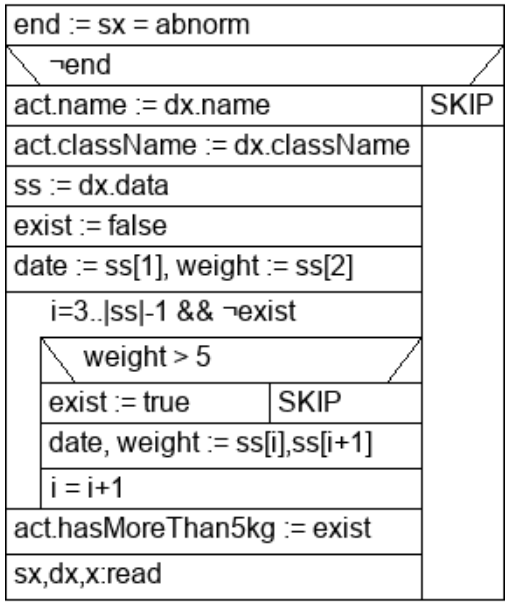
Contribution = **rec**(name: 𝕊,className: 𝕊,data: 𝕊)

StudentPaperData = **rec**(name: 𝕊,className: 𝕊,hasMoreThan5kg: 𝕃)

*Efnext =* ( x = x’ ꓥ dx = dx’ ꓥ sx = sx’ )

*Ufnext =* ( end = (sx’ = abnorm) ꓥ (¬end ⟶ (act.name = dx’.name

ꓥ act.className = dx’.className ꓥ (act.hasMoreThan5kg =

 ))

*A megvalósítás kódja:*

*ha ss egy istringstream*

*ss >> date >> weight*

*ss >> date >> weight*

Pesszimista lineáris keresés (eldöntésre)

# (2) részfeladat megoldása

## Főprogram terve

*A* = ( t : enor(𝕃), c : ℕ)

*Ef* = ( t = t’ )

*Uf =* ( c )

Megszámlálás

## Iskolai osztályok felsorolója

Felsoroló:

t:enor(𝕃) StudentPaperData = **rec**(name: 𝕊,className: 𝕊,hasMoreThan5kg: 𝕃)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 𝕃 \* | first() | next() | current() : 𝕃 | end() : 𝕃 |
| y : enor(StudentPaperData) dy : StudentPaperData act : 𝕃 end : 𝕃 | y.first()  next() | lásd külön | **return** act | **return** end |

Az enor(𝕃) „Iskolai osztályok felsorolója”-nak first() művelete meghívja az „y” adattagjának a first() műveletét, amely lépteti annak a felsorolását, és beállítja annak az „end” értékét. Ezt követően meghívja a next() műveletet.

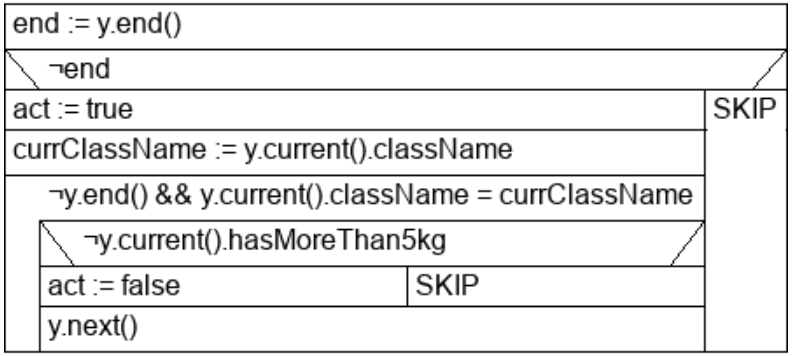
A next() művelet az „y” adattag „end” mezőjének függvényében állítja be a felsoroló „end” mezejét, hiszen a külső felsorolás is akkor ér véget, amikor a belső. Ezután egy lineáris keresést hajt végre, ami addig megy, amíg vagy véget nem ér a felsorolás vagy a belső felsorolt elem „className” mezője eltér a kezdeti értéktől. Ezen belül, ha talál a belső felsorolásban a „hasMoreThan5kg” mezőben hamis értéket, akkor a felsorolás „act” mezejébe hamis érték kerül, ha nem talál ilyet akkor pedig igaz.

Next Művelet:

*Anext* = ( y : enor(StudentPaperData), dy: StudentPaperData, act: 𝕃, end: 𝕃)

StudentPaperData = **rec**(name: 𝕊,className: 𝕊,hasMoreThan5kg: 𝕃)

*Efnext =* ( y = y’ ꓥ dy = dy’)

*Ufnext =* ( end = (y’.end()) ꓥ (¬end ⟶ akt ) ))

Lineáris keresés (eldöntésre)

# Tesztelési terv:

A megoldásban négy programozási tételt alkalmaztunk:

összegzés (kiválogatás),megszámlálás, lineáris keresés (eldöntésre), pesszimista lineáris keresés (eldöntésre)

A tesztesetekhez tartozó fájlok a ./tests/ mappában érhetőek el, a tesztek pedig az ./src/tests.h fájlban.

## (1) Feladat| Az összegzés (kiválogatás) tesztesetei

Minden tesztesethez 2 fájl tartozik:

Egy **bemeneti** (source) fájl: **Test%%S.txt**

Egy **várt eredmény** (expected) fájl: **Test%%E.txt**

A tesztelő összehasonlítja a bemeneti fájlokhoz tartozó kimeneteket a várt eredményeket tartalmazó fájlokkal és hiba esetén jelenti a különbségeket, illetve azok számát.

-Tanulók, akik legalább egyszer több, mint 5kg papírt hoztak

* **bemenet hossza** szerint és **eredmények száma** szerint:

1. nem létező bemeneti fájl
2. 0 tanuló 0 eredmény - üres fájl
3. 1 tanuló – 0 eredmény
4. 1 tanuló – 1 eredmény
5. 2 (több tanuló) - 0 eredmény
6. 2 (több tanuló) – 1 eredmény
7. 2 (több tanuló) – 2 eredmény

* kiválogatott **elemek elhelyezkedése** szerint (mindegyik több tanulós bemenet)

1. eredmény a bemenet elején
2. eredmény a bemenet közepén
3. eredmény a bemenet végén

* Bemeneti tanulók nevének hosszai szerint (2,több)

## (2) Feladat| A megszámlálás, lineáris keresés tesztesetei

Minden tesztesethez 1 fájl tartozik:

Egy **bemeneti** (source) fájl: **Test%%S.txt**

A tesztelő összehasonlítja a bemeneti fájlokhoz tartozó teszteredményeket a várt eredményekkel. Mivel a várt eredmény maga egy szám, ezért a tesztelőbe van belekódolva, nem pedig külön fájlban.

- Osztályok száma, ahol minden tanuló legalább egyszer több, mint 5 kg papírt hozott

* **Bemenet hossza** (osztályok száma) és **eredmény** (db) szerint

Az eredmények számának ellenőrzésébe van **beépítve a lineáris keresés** (eldöntésre) tesztelése.

1. Nem létező bemeneti fájl
2. 0 osztály,0 tanuló – üres fájl – eredmény: 0
3. 1 osztály, 1 tanuló – eredmény: 0
4. 1 osztály, 1 tanuló – eredmény: 1
5. 1 osztály, 2 (több) tanuló – eredmény: 0 (egyikre sem igaz)
6. 1 osztály, 2 (több) tanuló – eredmény: 0 (nem mindre igaz)
7. 1 osztály, 2 (több) tanuló – eredmény: 1 (mindre igaz)
8. 2 (több) osztály – több tanuló – eredmény: 0
9. 2 (több) osztály – több tanuló – eredmény: 1
10. 2 (több) osztály – több tanuló – eredmény: 2

## Közös|Pesszimista Lineáris (eldöntés) keresés tesztesetei

1 beolvasott sorban történő pesszimista lineáris keresés (eldöntésre) tesztjei. Ellenőrzi, hogy ha egy tanulóhoz több papírgyűjtési adat is tartozik, akkor van-e köztük 5kg-nál nagyobb.

Minden tesztesethez 1 fájl tartozik:

Egy **bemeneti** (source) fájl: **Test%%S.txt**

A tesztelő összehasonlítja a bemeneti fájlokhoz tartozó teszteredményeket a várt eredményekkel. A várt eredmény mindig vagy 0 – nincs 5kg-nál nagyobb - vagy 1 – van 5kg-nál nagyobb. Minden tesztfájlban maximum 1 ilyen adat lehet, és **annak a létezését, és különböző helyeit teszteljük.**

1. | eredmény: 0 – nincs 5kb-nál nagyobb adata
2. | eredmény: 1 – van, az első adat
3. | eredmény: 1 – van, a második (középső) adat
4. | eredmény: 1 – van, a harmadik (utolsó adat)