Gregorics Tibor **2. beadandó/0.feladat** 2012. január 11.
EHACODE.ELTE
gt@inf.elte.hu
0.csoport

Feladat

Egy szöveges állományban bekezdésekre tördelt szöveg található. Egy bekezdés egy vagy több nem üres sorból áll. A bekezdéseket üres sorok vagy az állomány eleje illetve vége határolja. Gondolatban sorszámozza meg a bekezdéseket és írja ki a konzolra azon bekezdések sorszámait, amelyek minden sorának legalább két szavában szerepel a 'w' betű, valamint írja ki az ilyen bekezdésben a 'w' betűt tartalmazó szavak és az összes szó arányát is. (A szövegben egyik szó sincs több sorra tördelve.).

Specifikáció

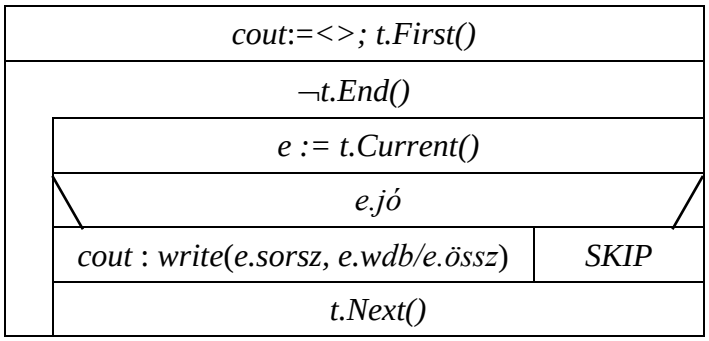
A feladat állapottere többféleképpen is felírható,

$A = (f : InFile(\mathbb{K}), cout : OutFile(\mathbb{N}, \mathbb{R}))$
$A = (f : InFile(Sor), cout : OutFile(\mathbb{N}, \mathbb{R}))$

de a megoldás szempontjából legjobb, ha egy olyan felsoroló objektumra fogalmazzuk meg, amely képes a bekezdések számunkra fontos adatait felsorolni.

$A = (t : Enor(Bekezdés), cout : OutFile(\mathbb{N}, \mathbb{R}))$
$Bekezdés = \mathbf{rec}(jó:\mathbb{L}, sorsz:\mathbb{N}, wdb:\mathbb{N}, össz:\mathbb{N})$
$Ef = (t = t')$
$Uf = (cout = \bigoplus_{\substack{e \in t' \\ e.jó}} < e.sorsz, e.wdb / e.össz >)$

Absztrakt program



Felsoroló

enor(Bekezdés)	*First(), Next(), Current(), End()*
f : *InFile*(*String*), *sor* : *String*, *st* : *Státusz* *akt* : *Bekezdés*, *vége* : $\mathbb{L}$	*First() ~ akt.sorsz:=0; st, sor, f : read; Next()* *Next() ~ ld. külön* *Current() ~ akt* *End() ~ vége*

A *Next()* műveletnek az alábbi feladatot kell megoldania:

Adott egy sorokra tördelt szöveges állomány ahonnan már kiolvastuk az első sort (ez van a *sor* változóban, ha sikeres volt ez a korábbi olvasás, mert különben *st=abnorm*) és soron következő bekezdését kell feldolgoznunk. Adott a korábbi bekezdés sorszáma (*akt.sorsz*). Először átolvassuk a bekezdés előtti üres sorokat. Ha nem találunk ezek után egy nem üres sort, akkor a *vége* változót igazra állítjuk (nincs több bekezdés). Ha találunk, akkor a *vége* változót hamisra állítjuk, az *akt.sorsz* értékét eggyel növeljük, és az *akt* többi mezőjét az itt kezdődő bekezdés alapján kitöltjük: megszámoljuk a szavakat, a '*w*' betűs szavakat, továbbá vizsgáljuk, hogy minden sorban volt-e legalább két '*w*' betűs szó. Mindhárom vizsgálathoz soronként kell az adott sorban megszámolni külön a szavakat és külön a '*w*' betűs szavakat. A bekezdés feldolgozása végén vagy a bekezdést követő üres sort olvassuk be utoljára (ez kerül a *sor* változóba), vagy elértjük az állomány végét (*st=abnorm*).

$$A^{Next} = (f: infile(String),\ sor: String,\ st\ Státusz,\ vége:\mathbb{L},\ akt:Bekezdés)$$

$$Ef^{Next} = (\ f=f^1 \wedge sor=sor^1 \wedge st=st^1 \wedge akt=akt^1)$$

$$Uf^{Next} = (\ st^2, sor^2, f^2 = \underset{sor\in(sor^1,f^1)}{\textbf{select}} (st = abnorm \vee sor \neq üres) \wedge$$

$$(vége = (st^2 = norm)) \wedge$$

$$(vége \rightarrow$$

$$akt.sorsz = akt^1.sorsz+1$$

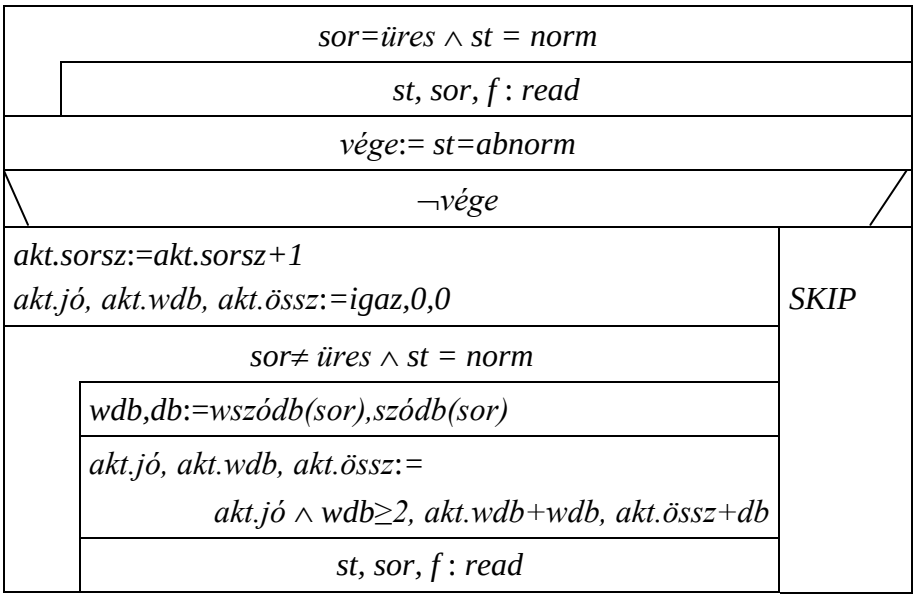
$$\wedge\ akt.jó, st, sor, f = \underset{sor\in(sor^2,f^2)}{\overset{sor\neq üres}{\wedge}} (wszódb(sor) \geq 2) \wedge$$

$$\wedge\ akt.össz, st, sor, f = \underset{sor\in(sor^2,f^2)}{\overset{sor\neq üres}{\Sigma}} (szódb(sor)) \wedge$$

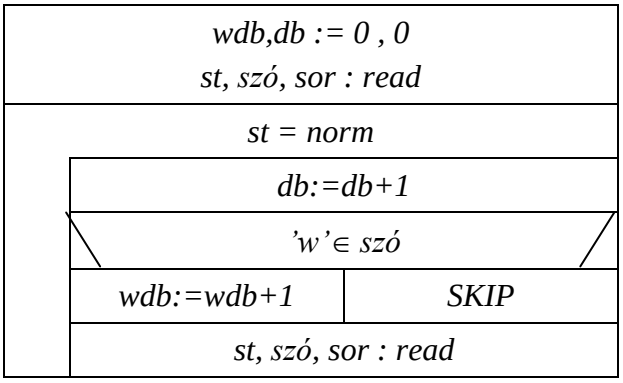
$$\wedge\ akt.wdb, st, sor, f = \underset{sor\in(sor^2,f^2)}{\overset{sor\neq üres}{\Sigma}} (wszódb(sor))\ ))$$

ahol $szódb(sor) = \sum_{szó\in sor} 1$ és $wszódb(sor) = \sum_{\substack{szó\in sor \\ 'w'\in szó}} 1$ számlálásai egy sor szavanként olvasására épülnek, és egyetlen ciklusba összevonhatók. Összevonható egy ciklusba az utófeltételben vázolt három összegzés is.

Next()



wdb,db:=wszódb(sor),szódb(sor)



Implementáció

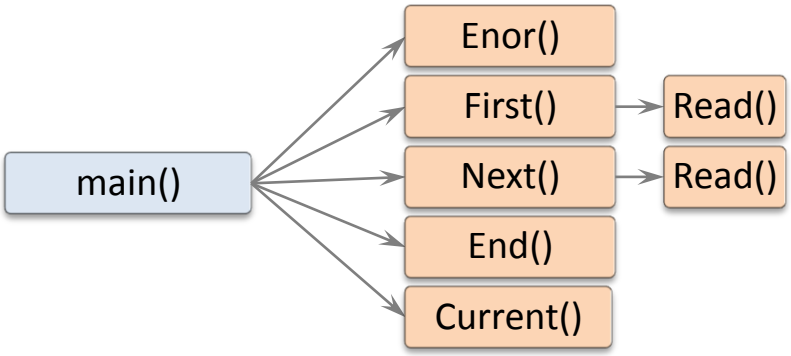
Program váz

A program több állományból áll: `main.cpp, enor.h, enor.cpp`.

main.cpp	**enor.h**	**enor.cpp**

`int main()`	`struct Bekezdes`
	`enum Status`
	`class Enor`
	`Enor()`
	`void First()`
	`Bekezdes Current()`
	`bool End()`
		`void Read()`
		`void Next()`

Függvények kapcsolódási szerkezete



Felsoroló osztálya

```
class Enor{
    private:
        std::ifstream f;
        std::stringstream sor;
        Status st;
        Bekezdes akt;
        bool vege;
        void Read();

    public:
        Enor(const std::string &str);
        void First() { akt.sorsz = 0; Read(); Next();}
        void Next();
        Bekezdes Current() const { return akt;}
        bool End() const { return vege;}
};
```

A szöveges állomány soronként olvasását a `getline(f,sor)` utasítás végzi, amelyet beágyazunk a `Read()` metódusba úgy, hogy a sort `stringstream` típusú adatként adja vissza, illetve beállítsa az olvasás *st* státuszát.

A `stringstream` típusú sor szavainak olvasása a >> operátorral történik. Üres egy sor, ha sztringként értelmezve az üres sztring.

Egy szóban a 'w' betű keresését a `string` típus `find()` függvényével végezzük. A megvalósításban a *wszódb()* és *szódb()* függvények nem kerültek önálló alprogramba.

Tesztelési terv

A megoldásban hat helyen alkalmaztunk programozási tételt, amelyek három szintre tagolódnak: bekezdések, sorok, szavak szintjeire.

A. Bekezdések szintjén intervallum és összegzés (kiválogatás) tesztesetei:

1. Üres állomány.
2. Csak üres sorokat tartalmazó állomány.
3. Egyetlen (keresett tulajdonságú) bekezdést tartalmazó állomány.
4. Több (keresett tulajdonságú) bekezdést tartalmazó állomány.

5. Első bekezdés keresett tulajdonságú, a többi nem.
6. Utolsó bekezdés keresett tulajdonságú, a többi nem.
7. Több keresett tulajdonságú bekezdés

B. Egy bekezdéshez tartozó sorok szintjén intervallum és három összegzés (összeéselés és két összeadás) egybefoglalt tesztesetei:

8. Egyetlen nem üres sorú (keresett tulajdonságú) bekezdés, előtte és utána üres sorok vagy az állomány vége.
9. Több soros bekezdésből álló állomány, előtte és utána üres sorok vagy az állomány vége.

10. Több soros bekezdésből álló állomány, ahol minden sorban kettő vagy annál több ’w’-t tartalmazó szó van.
11. Több soros bekezdésből álló állomány, ahol csak az első sorban nincs kettő vagy annál több ’w’-t tartalmazó szó.
12. Több soros bekezdésből álló állomány, ahol csak az utolsó sorban nincs kettő vagy annál több ’w’-t tartalmazó szó.
13. Több soros bekezdésből álló állomány, ahol csak egy közbülső sorban nincs kettő vagy annál több ’w’-t tartalmazó szó.
14. Több soros bekezdésből álló állomány, ahol egyik sorban sincs kettő vagy annál több ’w’-t tartalmazó szó.

C. Egy sorhoz tartozó szavak szintjén intervallum és a két összegzés (összegzés és számlálás) egybefoglalt tesztesetei:

15. Egy bekezdés egyetlen sorában nincs szó.
16. Egy bekezdés egyetlen sorában egyetlen szó van.
17. Egy bekezdés egyetlen sorában több szó van.

18. Egy bekezdés egyetlen sorában nincs ’w’-t tartalmazó szó.
19. Egy bekezdés egyetlen sorában csak az első a ’w’-t tartalmazó szó.
20. Egy bekezdés egyetlen sorában csak az utolsó a ’w’-t tartalmazó szó.
21. Egy bekezdés egyetlen sorában egynél több ’w’-t tartalmazó szó van.
22. Egy bekezdés egyetlen sorában minden szó ’w’-t tartalmaz.

Az egyes alprogramok fehérdoboz tesztjét a fenti tesztesetek lefedik, ennél a fogva az egyedi felsorolót megvalósító osztály metódusait is. (Nincs szükség a metódusok variációs tesztjére, hiszen a felsoroló műveleteit csak meghatározott sorrendben lehet hívni, amit a rá támaszkodó programozási tétel szerkezete biztosít.)