Programozás beadandó feladat:  
2. beadandó / 7. feladat

Készítette: Podlovics Péter Dávid  
Neptun-kód: D0W5T4  
E-mail: peter.d.podlovics@gmail.com

Gyakorlatvezető neve: Veszprémi Anna/Hudoba Péter

2016. május 6.

.

Contents

[Feladat: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516147)

[Megoldási terv: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516148)

[Specifikáció: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516149)

[Visszavezetés: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516150)

[Maximum kiválasztás: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516151)

[Megszámlálás: 3](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516152)

[Implementáció: 5](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516153)

[Az adattípusok megvalósítása: 5](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516154)

[Bemenő adatok formája: 5](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516155)

[Program váz: 6](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516156)

[Tesztelési terv: 8](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516157)

[Fekete doboz tesztesetek: 8](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516158)

[Érvényes tesztesetek: 8](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516159)

[Érvénytelen tesztesetek: 10](file:///C:\ELTE\II.%20félév\Programozás\I.%20Beadandó\Beadando%20I\dokumentacio.docx#_Toc446516160)

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. Melyik a leggazdagabb bekezdés, azaz hányadik az a legalább három soros bekezdés, ahol a legnagyobb a szavak számának és a sorok számának hányadosa? (A szövegben egyik szó sincs több sorra tördelve.) Tipp: Készítsen olyan felsorolót, amelyik a bekezdéseket, pontosabban azoknak számunkra fontos tulajdonságát sorolja fel!

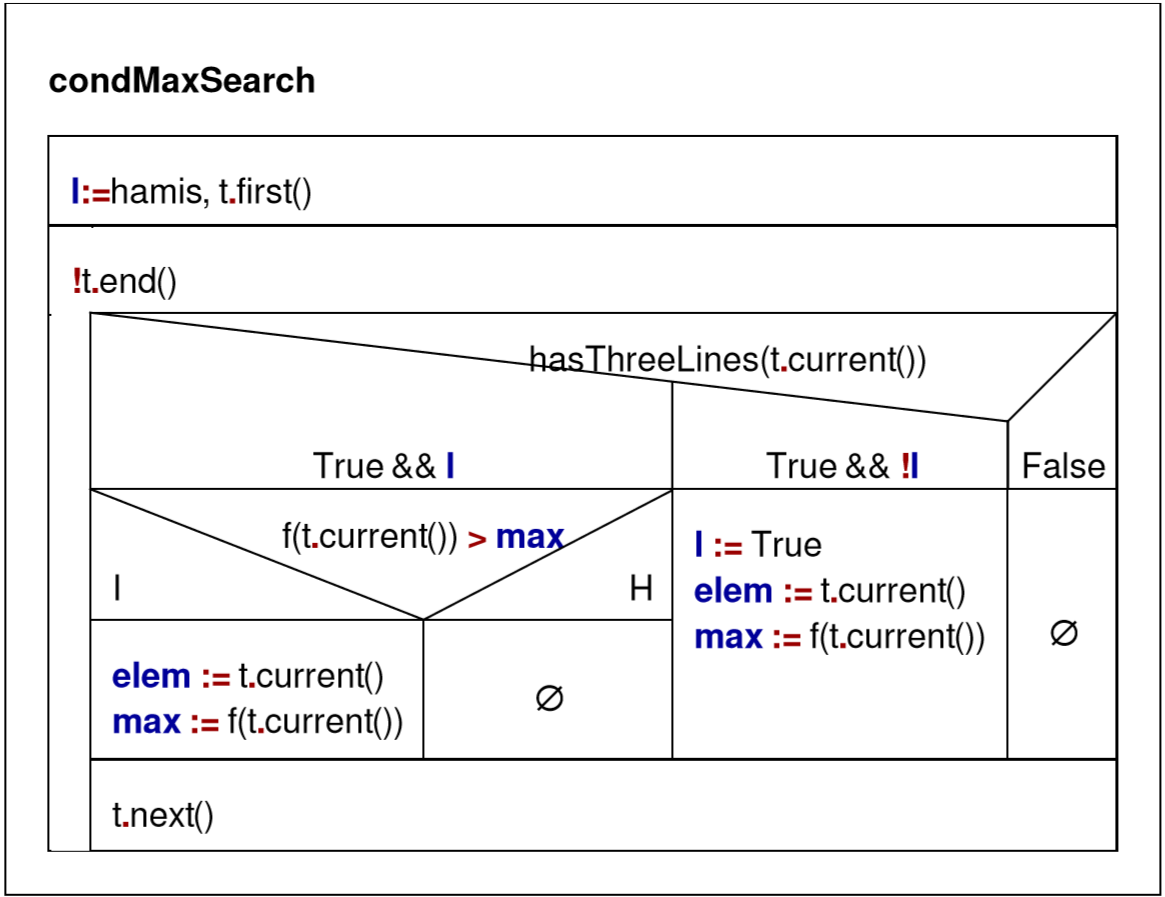
Megoldási terv:

Specifikáció:

Visszavezetés:

Feltételes maximum keresés:

A maximum keresést kiválasztást a condMaxSearch függvény fogja végrehajtani

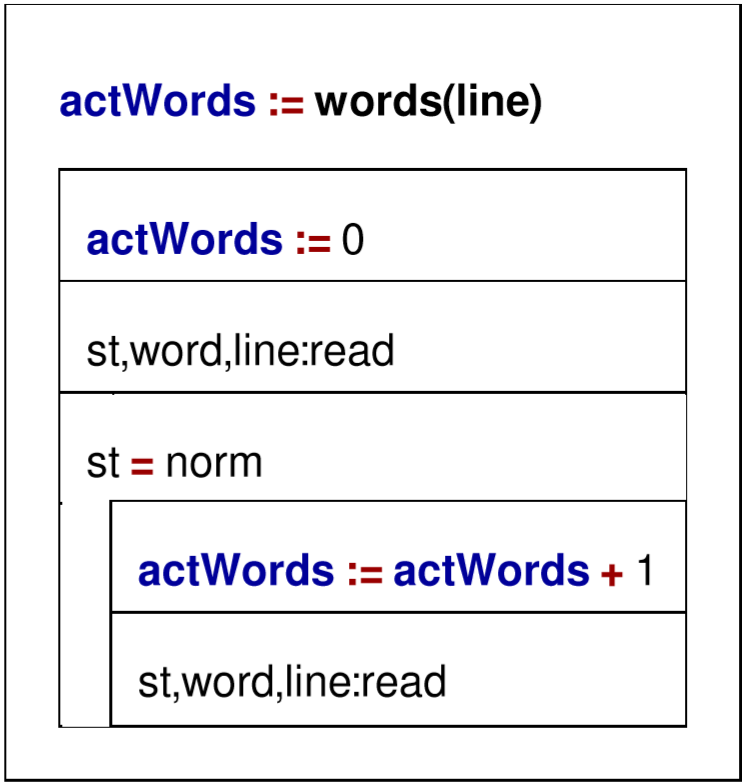
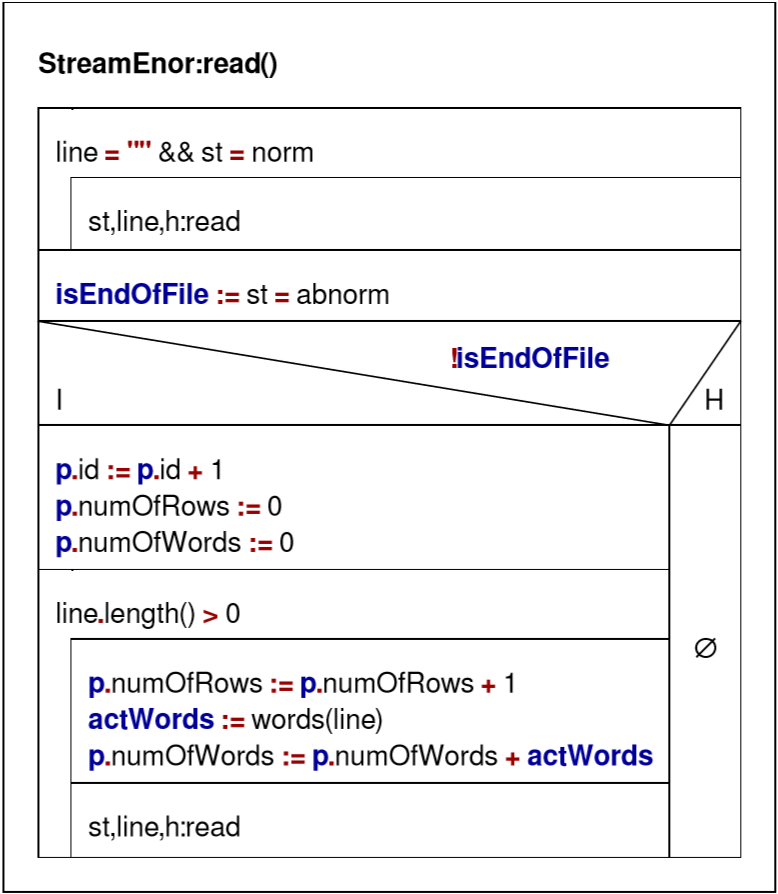


StreamEnor:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StreamEnor(E) | First() | Next() | l:= End() | e := Current() |
| f:infile(E)  sf : Status  df : E  read | Sf,df,read | Sf,df,read | Sf = abnorm | e := df |

A read függvény eltér a már megszokott soronkénti olvasástól. Ez a függvény egyszerre egy teljes Paragraph-ot beolvas az adott streamről. Ezt a következő képpen teszi:

Beolvassa az összes üres sort a stream elejéről, majd elkezdi olvasni az aktuális paragrafus sorait, és ezeket fogja feldolgozni. Mikoözben olvassa a sorokat, folyamatosan módosítja a jelenlegi Paragraph numOfWords és numOfRows mezőit.



Implementáció:

Bemenő adatok formája:

A bemenő adatokat kétféle képpen adhatjuk meg: konzolról való beolvasás során, illetve szöveges állományból való beolvasással. A konzolról történő adatbevitel során a bemenet megadása utána end\_of\_file jelet kell küldenünk a programnak (ez operációs rendszerenként eltérő lehet)

Program váz:

A függvények kapcsolódási rendszere:

A Paragraph-ok beolvasását egy overload-olt operator>>() végzi. A soronkénti olvasásért a getline() függvény felelős. Egy sort egy stringstream-ben tárolunk, ezt dolgozzuk fel.

A program állományai:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Main.cpp** | **Menu.cpp** | **Paragraph.cpp** | **Enor.h** | **Enor.hpp** | **Menu.h** | **Paragraph.h** |
| Struct QueryResult,  Bool hasThreeLines(),  Bool compareParagraphs(),  QueryResult condMaxSearch() | Int checkedReadLIne(),  Void displayMenu,  Bool checkCommand,  Int readCommand,  Void quit() | Istream& operator>>,  Ostream& operator<< | Enum Status,  Struct StreamEnorException,  Class Enor,  Class StreamEnor | StreamEnorException(),  Ostream& operator<<,  StreamEnor() | A menu.cpp függvényei | Class Paragraph |

Tesztelési terv:

Fekete doboz tesztesetek:

Érvényes tesztesetek:

1. **A bekezdések szintjén (max. ker)**
   1. *Intervallum hossza szerint:*
      1. Üres állomány:

Bemenet: in1.txt

Kimenet: NOT FOUND

* + 1. Csak üres sorokat tartalmazó állomány:

Bemenet: in2.txt

Kimenet: NOT FOUND

* + 1. Egyetlen (feltételnek megfelelő) paragrafust tartalmazó állomány

Bemenet: in3.txt

Kimenet:

id: 2

Number of rows: 3

Number of words: 6

Ratio: 2

* + 1. Több (feltételnek megfelelő) paragrafust tartalmazó állomány

Bemenet: in4.txt

Kimenet:

id: 2

Number of rows: 3

Number of words: 6

Ratio: 2

* 1. *Tétel szerint:*
     1. Az első elem maximális:

Bemenet: in5.txt

Kimenet:

id: 1

Number of rows: 4

Number of words: 6

Ratio: 1.5

* + 1. Az utolsó elem maximális:

Bemenet: in6txt

Kimenet:

id: 2

Number of rows: 4

Number of words: 6

Ratio: 1.5

* + 1. Minden elem maximális:

Bemenet: in7.txt

Kimenet:

id: 1

Number of rows: 4

Number of words: 6

Ratio: 1.5

1. **Egy bekezdéshez tartozó sorok szintjén (megszámlálás):**
   1. *Intervallum hossza szerint:*
      1. *Egy sort tartalmaz:*

Bemenet: in8.txt

Kimenet: NOT FOUND

* + 1. *Több sort tartalmaz:*

Bemenet: in9.txt

Kimenet:

id: 2

Number of rows: 4

Number of words: 6

Ratio: 1.5

1. **Egy sor szintjén (Megszámlálás):**
   1. *Intervallum hossza szerint:*
      1. Egy szó van:

Bemenet: in10.txt

Kimenet:

id: 1

Number of rows: 7

Number of words: 7

Ratio: 1

* + 1. Több szó van:

Bemenet: in11.txt

Kimenet:

id: 2

Number of rows: 4

Number of words: 21

Ratio: 5.25