

17-es feladat

Szöveg-átalakító

Szöveges fájl részeinek kiemelése

Programozás módszertan 3 beadandó

Kiss Endre Farkas

kissfarkas@t-online.hu

TARTALOM

FELHASZNÁLÓI KÉZIKÖNYV:	3
A PROGRAM FELADATA	3
KÖVETELMÉNYEK.....	3
<i>Operációs rendszer</i>	3
<i>Hardver</i>	3
INDÍTÁS	3
A PROGRAM HASZNÁLATA	4
<i>Kezelőfelület</i>	4
<i>Szöveg betöltése</i>	4
<i>Szöveg átalakítása</i>	4
<i>Szöveg kimentése</i>	5
<i>Segítség</i>	5
<i>Munka befejezése</i>	5
FEJLESZTŐI DOKUMENTÁCIÓ	6
FELADAT	6
SPECIFIKÁCIÓ.....	6
FŐBB ÁLLOMÁNYOK.....	7
TESZTELÉS.....	8
<i>Első tesztet jegyzetombben</i>	8
<i>Az első tesztet az alkalmazásban megformázva</i>	8
<i>Második tesztet jegyzetombben:</i>	9
<i>Az második tesztet az alkalmazásban megformázva</i>	9
<i>Harmadik tesztet jegyzetombben</i>	9
<i>Az harmadik tesztet az alkalmazásban megformázva</i>	9
TOVÁBBFEJLESZTÉS LEHETŐSÉGEI	10
ALGORITMUS	10
<i>A használt változók</i>	10
<i>Funkcionálisan elkülönülő kódrészek:</i>	12
FORRÁSKÓD:.....	14
<i>„Unit 1” unit</i>	14
<i>„Feldolgoz” unit:</i>	15
SZERZŐ ADATAI.....	19

FELHASZNÁLÓI KÉZIKÖNYV:

A program feladata

A program, amely egy szöveg fájl egyes részeit képes kiemelni (ritkítva, illetve aláhúzva írni). A kiemelendő részeket a szövegben #-, illetve \$-jelek közé kell tennünk. Az aláhúzás miatt a konvertált szöveg a kimeneti fájlban két sorosan jelenik meg.

(Ha a fenti jelek szerves részei a szövegnek, akkor duplázni kell őket!) Pl.:

BEMENET	KIMENET
Készítsen #programot#, amely egy \$szöveg\$ fájl egyes \$részeit képes ...	Készítsen p r o g r a m o t , amely egy szöveg ----- fájl egyes részeit képes ... -----
... a szövegben ##-, illetve \$\$- jelek közé a szövegben #-, illetve \$- jelek közé ...

követelmények

Operációs rendszer

A program elfut bármilyen Win9x, illetve WinNT kompatibilis rendszer alatt. (pl. *Windows 95, 98, Me, XP*).

Hardver

Az alkalmazás Windows rendszereket futtatni képes rendszerek esetén nem támaszt extra igényeket gépünkkel szemben, régebbi rendszerek esetén minimális követelményként kell tekintenünk az alábbi konfigurációra:

- 300 Mhz CPU
- 128 Mb RAM
- SVGA vezérlő
- 300 Mb HDD
- 14' monitor

Indítás

Az iskolai sportjelentkezéseket nyilvántartó program a mellékelt CD lemezen a főkönyvtárban lévő atalakito.exe fájljal indítható.

Hibalehetőségek:

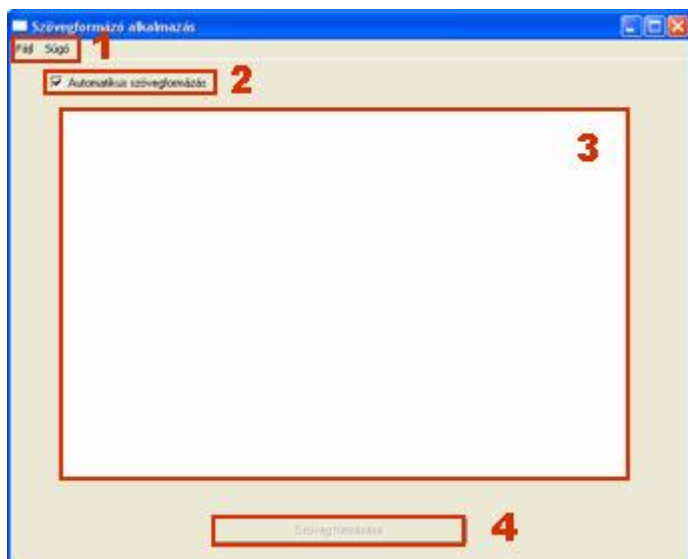
A program fejlesztése során igyekeztem maximális körültekintéssel eljárni, azonban egyes speciálisan formázott fájlok esetén nem zárható ki a leírás szerinti funkcionalitástól eltérő működés.

A program használata

Indítás után a kezdőképernyőhöz jutunk.

Kezelőfelület

Az alkalmazás kezelőfelülete négy főbb részből áll.



1. Menüsor

Szövegállományok betöltéséhez, kimentéséhez, és információk megjelenítéséhez

2. Automatikus formázás beállító

Megadja, hogy a betöltendő szöveg átalakítva, vagy anélkül kerüljön betöltésre

3. Szövegterület

Az átalakított vagy átalakítás előtti szöveg itt jelenik meg.

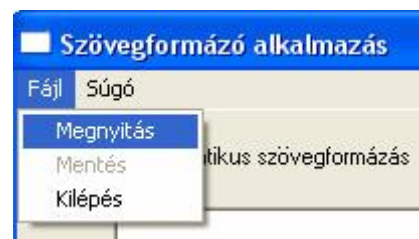
4. Átalakító gomb

Ha aktív, a szövegterületben lévő szövegnek ad, vagy szünteti meg a formázottságát.

Szöveg betöltése

Szöveget az alkalmazásba a Fáj > Megnyitás menüpont alatt tölthetünk be.

Ha rákattintunk, akkor a Windows szokványos fájlmegnyitó ablaka nyílik meg, ahol .txt kiterjesztésű szövegállományok megnyitására nyílik lehetőségünk.



Miután ezt megtesszük, a megnyitott állomány tartalma megjelenik

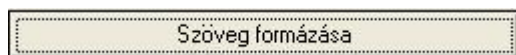
a Szövegterületben. Az, hogy formázott, vagy a speciális formázó karaktereket is magában foglaló módon történik-e meg attól függ, hogy megnyitás előtt bejelölte a az Automatikus szövegformázás jelölőnégyzetet.

Szöveg átalakítása

A betöltött szöveg utólagos átalakítására a kezelőfelület alján található Átalakító gombbal nyílik lehetőségünk. Ez a gomb csak akkor aktív, ha már van betöltött szöveg a Szövegterületben.

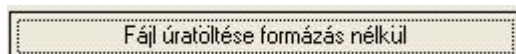
Amíg ez nem történik meg, nem tudunk rákattintani.

Az átalakító gomb szövege kétféle lehet.



Amikor a betöltött szöveg nincs megformázva, vagyis magában foglalja a speciális formázó karaktereket is.

Ha ekkor kattintunk a gombra, akkor a Szövegterületen lévő szövegből eltűnnek a speciális formázó karakterek, és a szöveg formázva (a megfelelő helyen ritkítva, és aláhúzva) jelenik meg.

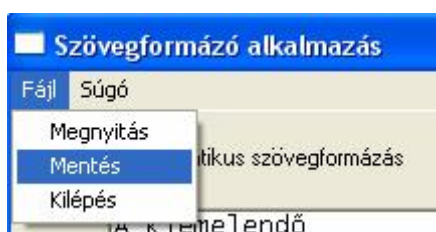


Amikor a betöltött szöveg már formázva van a Szövegrészben.

Ha ekkor kattintunk a gombra, akkor a Szövegterületen lévő szöveg formázottsága megszűnik, és aláhúzott, illetve ritkított szöveg helyett a megfelelő helyeken megjelennek a speciális formázó karakterek.

Szöveg kimentése

A szöveg kimentését a Fájl > Mentés menüpont alatt tudjuk elvégezni.



Ez a menüpont csak akkor aktív, ha már van betöltött szöveg a Szövegterületen. Ha még nem töltöttünk be szöveget, akkor erre a menüpontra még nem tudunk kattintani.

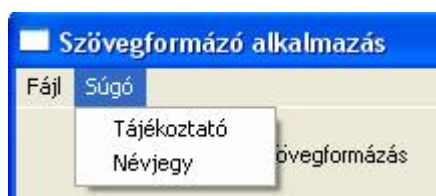
Ha már van kimentendő szövegünk, akkor a menüpont hatására a Windows megszokott Fájl mentés ablaka nyílik meg, ahol

kimenthetjük a szövegmező tartalmát tetszőleges .txt kiterjesztésű szövegfájlba.

Ilyenkor mindig a Szövegterület aktuális állapota kerül kimentésre.

Ha formázott a szövegünk, akkor a formázott verzió, ha nincs formázva, akkor a formázatlan verzió.

Segítség



A programról használat közben információt a Súgó menü segítségével kaphatunk.

A Tájékoztató menüpont rövid, tömör összefoglalását jeleníti meg jelen használati utasításnak.

A Névjegy menüpont rövid információt jelenít meg a program készítőjéről, és a fejlesztés dátumáról.

Munka befejezése

Kilépni az alkalmazásból a Fájl > Kilépés menüponttal tudunk.

FEJLESZTŐI DOKUMENTÁCIÓ

Feladat

Olyan program készítése, amely egy szöveg fájl egyes részeit képes kiemelni (ritkítva, illetve aláhúzva írni). A kiemelendő részeket a szövegben #-, illetve \$-jelek közé tettük. Az aláhúzás miatt a konvertált szöveg a kimeneti fájlban két sorosan kell, hogy megjelenjen. (Ha a fenti jelek szerves részei a szövegnek, akkor duplázni kell őket!)

Specifikáció

Be: SzövegF \in Szöveg

Ki: kiSzöveg \in Szöveg

Ef: hossz (SzövegF) > 1

Uf: keresztSzám (k, sor) = <SzövegF.sorok(sor)-ban k-ik karakterig '#' jelek között levő karakterek száma>

\wedge sorokSzáma (kiSzöveg) = 2 * sorokSzáma (SzövegF)

$\wedge \forall \text{ sor } [1.. \text{sorokSzáma (SzövegF)}] :$

alahuzasSzam = jelekSzáma ('\$', SzövegF.sorok(sor))

$\wedge \forall i [1.. \text{alahuzasSzam}] : \text{alahuzasHely}_i = \text{'\$'} \text{ jel } i\text{-edik előfordulási helye SzövegF.sorok(sor)-ban}$

$\wedge \forall j [1.. \text{hossz(SzövegF.sorok(sor))}] :$

Ha SzövegF.sorok(sor)_j \neq '\$'

\wedge SzövegF.sorok(sor)_j \neq '#':

$\Rightarrow \text{kiSzöveg.sorok((sor*2)-1)}_{j + \text{keresztSzám}(j, \text{sor})} = \text{SzövegF.sorok(sor)}_j$

\wedge Ha keresztSzám(j, sor) \nmid 2: $\text{kiSzöveg.sorok}((\text{sor}*2)-1)_{j + \text{keresztSzám}(j, \text{sor}) - 1} = \text{'szóköz'}$

Ha $j > \text{alahuzashely}_i \wedge j < \text{alahuzashely}_{i+1}$

$\Rightarrow \text{KiSzöveg.sorok(sor * 2)}_j = \text{'kötőjel'}$

egyébként:

$\text{kiSzöveg.sorok(sor * 2)}_j = \text{'szóköz'}$

Főbb állományok

- **atvalto.exe**
A program indítása történik a segítségével.
- **tejekoztato.dat**
Az alkalmazás innen olvassa be a felhasználói kézikönyv röviden összefoglalt változatát.

A forras könyvtárban:

- **beadando.lpi**
Az alkalmazás Lazarus Projekt állománya, mely leírja, hogy mely állományok képezik a Projekt részét.
- **beadando.lpr**
A projekt leírást egészíti ki. Azt írja le, hogy az alkalmazás indításakor miket kell inicializálni.
- **feldolgoz.lfm**
Az alkalmazás fő űrlapjának leírása
- **feldolgoz.pas**
Feldolgoz modul. Az alkalmazás fő űrlapjának működését írja le.
- **unit1.lfm**
Az tájékoztató űrlap (ablak) leírása
- **unit1.pas**
Unit1 modul. A tájékoztató űrlap (ablak) működését írja le.

A forras/includes könyvtárban:

- **ujsor.inc**
Kódrészlet, mely azt határozza meg, hogy egy formázandó szöveg formázása közben > mi történjen sorvége, vagy fájlvége felérésekor.
- **formazoKarakterek.inc**
Kódrészlet, mely azt határozza meg, hogy egy formázandó szöveg feldolgozása közben > miként reagáljon a \$, illetve # speciális karakterekre a program.

- [karakterSorVegere.inc](#)

Kódrészlet, mely azt határozza meg, hogy egy formázandó szöveg feldolgozása közben > milyen módon írja ki a soron következő karaktert a kimenetbe.

A doku könyvtárban:

- [atvalto.doc](#), [atvalto.pdf](#)

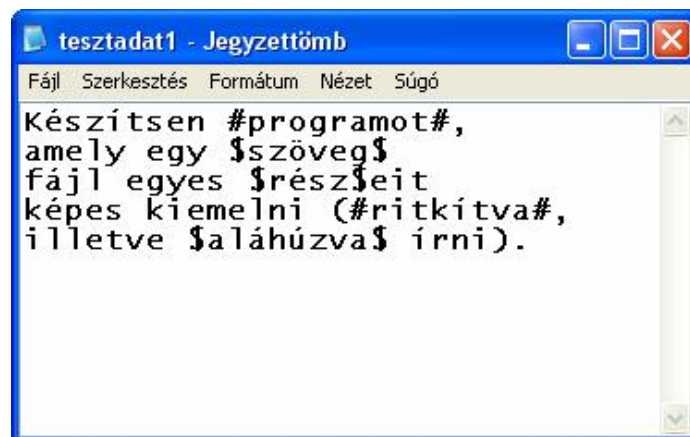
Jelen alkalmazás-dokumentáció Microsoft Word illetve Adobe Acrobat formátumban.

Tesztelés

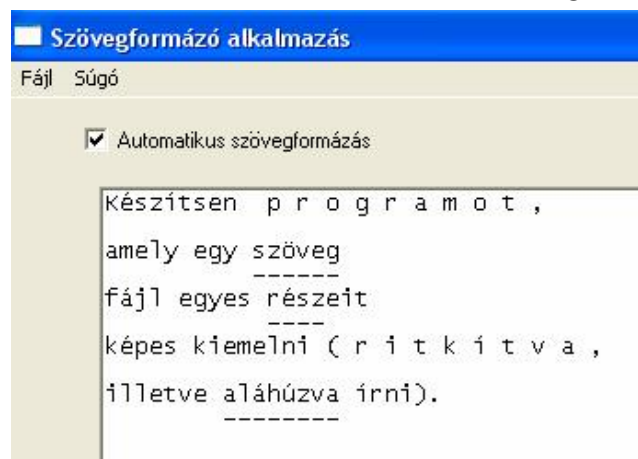
Az alkalmazás működésének teszteléséhez három teszt szöveget szerkesztettem a Windows XP beépített jegyzetömbjével.

A tesztesetek nagy része több-kevesebb átalakítással a követelmény-leírás szövegéből lett átemelve.

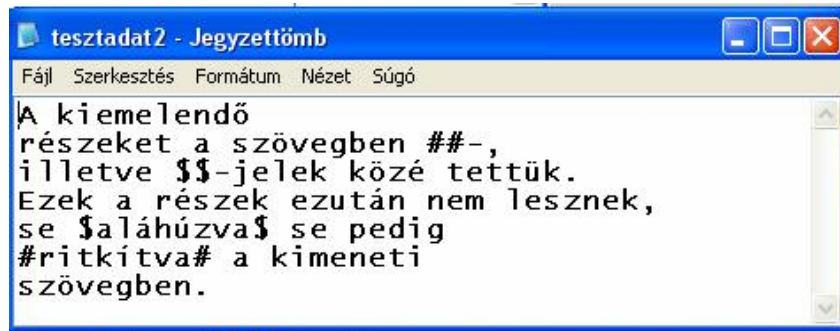
Első teszteset jegyzetömbben



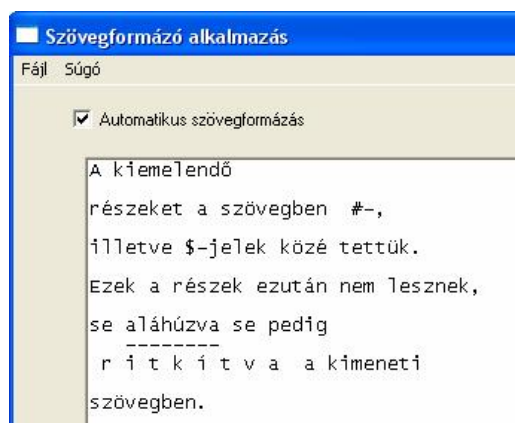
Az első teszteset az alkalmazásban megformázva



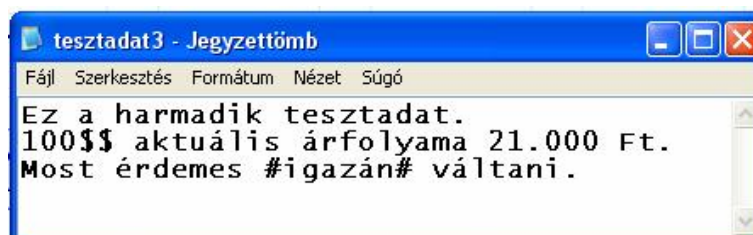
Második tesztet jegyzetömbben:



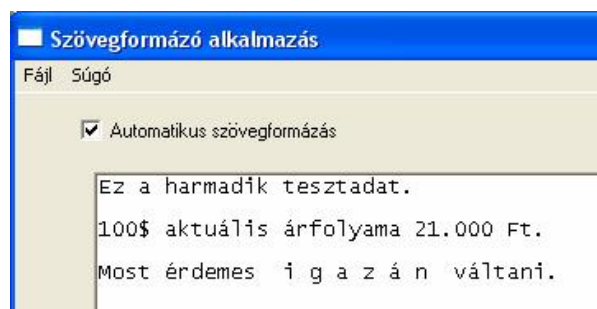
Az második tesztet az alkalmazásban megformázva



Harmadik tesztet jegyzetömbben



Az harmadik tesztet az alkalmazásban megformázva



Mivel a fenti három esetben nagyrészt lefedtük a formázó karakterek variálhatóságának a lehetőségeit, ezért a tesztelést befejezettnek tekinthetjük.

Ennek ellenére nincs kizárva, hogy a teszt során nem érintett, speciális kialakítású állományok esetén hibával kellene szembenéznünk.

Továbbifejlesztés lehetőségei

Az alkalmazást tovább lehetne fejleszteni úgy, hogy a felhasználónak lehetőséget biztosítunk arra, hogy a betöltött szövegeket futásidőben átszerkeszthessék akár normál, akár formázott módban. Nagyban megdobná a program használati értékét, ha ezt a szerkesztési funkciót a menüsor alatt elhelyezett formázó ikonsorral oldanánk meg, mely ilyen formán egy hagyományos WYSIWYG szövegszerkesztőhöz hasonlóan elrejtene a felhasználó előtt a formázott szöveg speciális karaktereit. A fentiekén túl nagyban növelné az alkalmazás üzembiztonságát, ha megoldanánk, hogy futtatás közben egy speciális átmeneti állományba kerüljön elmentésre a betöltött szövegállomány pillanatnyi állapota. Így egy esetleges program-vagy hardverhiba esetén ebből a biztonsági állományból helyreállíthatóak lennének azon adatok, melyek különben a hibával elvesztek volna.

Algoritmus

A használt változók

NÉV	HELY, TÍPUS	FUNKCIÓ
maxAlahuzas	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	Konstans, mely az program által egy sorban maximálisan kezelhető aláhúzások számát határozza meg.
szovegF	Feldolgoz modul, formaBeolvas Eljárás <i>Szöveges állomány</i>	A felhasználó által beolvasott szöveg.
alahuzasHely	Feldolgoz modul, formaBeolvas Eljárás Tömb 1..maxAlahuzas] : egész számokból	Ebben raktározzuk el, hogy a beolvasott szöveg az előző, illetve az aktuálisan feldolgozott sorban hányadik karakternél tartalmaz feldolgozandó (tehát nem duplán írt) aláhúzás-jelet.
sor	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	A beolvasott szöveg hányadik sorát dolgozzuk fel éppen.
j	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	Ciklusváltozó. Aláhúzás-sor írásakor ez jelöli, hogy a szöveg hányadik oszlopánál j éppen.
alahuzasSzam	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	Azt tároljuk benne, hogy az utoljára feldolgozott sorban hány darab aláhúzás-hely (kezdeti, és végpont) van.

NÉV	HELY, TÍPUS	FUNKCIÓ
karakterHely	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	Azt jelzi, hogy formázott szövegnél az aktuális soron belül hányadik karakterhelynél kell megjeleníteni a soron következő karaktert. Ritkított szöveg, illetve speciális karakterek esetén korrekciós tényezővel módosul. (Ha ki kell hagyni egy karaktert, csökken, ha ritkítani kell, karakterenként duplán nő).
alaHuzlr	Feldolgoz modul, formaBeolvas Eljárás <i>Egész szám</i>	Aláhúzás kirajzolásakor ez a változó jelzi, hogy az aláhúzások adatait jelző alahuzasHely tömb hány elemét dolgoztuk fel eddig.
ritkit	Feldolgoz modul, formaBeolvas Eljárás <i>Logikai változó</i>	Szöveg feldolgozása közben az aktuálisan feldolgozott helyen éppen ritkított-e a szöveg.
alahuz	Feldolgoz modul, formaBeolvas Eljárás <i>Logikai változó</i>	Szöveg feldolgozása közben az aktuálisan feldolgozott helyen éppen aláhúzott-e a szöveg.
ujsor	Feldolgoz modul, formaBeolvas Eljárás <i>Logikai változó</i>	Ha a szöveg feldolgozása folyamán elérjük a sor végét, ez jelzi, hogy a soron következő sorba aláhúzás, vagy egy új sor szöveg jön-e
c	Feldolgoz modul, formaBeolvas Eljárás <i>Karakter változó</i>	Bemeneti szöveg olvasása közben az aktuálisan beolvasott karakter

Funkcionálisan elkülönülő kódrészek:

„ujsor” rész:

- Feladata, hogy egy sor esetén kiírjon egy üres sort. Ebbe a sorba fog kerülni az esetleges aláhúzás.

```

ha ujsor vagy fájlvége(szovegF) akkor
    alahuz := hamis
    alahuzIr := 1

    ciklus j:=1-től (alahuzasHely[alahuzasSzam])-ig
        ha (j= alahuzasHely[alahuzIr]) akkor
            alahuz := nem (alahuz)
            alahuzIr := alahuzIr+1
        elágazás vége
        ha (alahuz) akkor
            Kimenet[sor] := Kimenet[sor]+ '-'
        különben
            Kimenet[sor] := Kimenet[sor]+ ' '
        elágazás vége
    ciklus vége
    vanAlahuzas := hamis
    sor:=sor+1
    ciklus j:=1-től alahuzasSzam+1-ig
        alahuzasHely[j] := 0
    ciklus vége
    alahuzasSzam := 0
    karakterHely := 0
    elágazás vége

```

„formazoKarakterek” rész:

- Feladata, hogy kezelje a szövegben előforduló sormázó karaktereket (\$ és #).

```

ha c = '#' akkor
    ha ritkit akkor karakterhely := karakterhely-1
    ritkit := nem(ritkit)
    elágazás vége

ha (elozo<>'$') és (c='$') akkor
    karakterhely := karakterhely - 1
    alahuzasSzam := alahuzasSzam+1
    alahuzasHely[alahuzasSzam] := karakterHely
    vanAlahuzas := igaz
    elágazás vége

ha (elozo='$') és (c='$') akkor
    alahuzasSzam := alahuzasSzam-1
    feltétel vége

```

„karakterSorVegere” rész:

- Feladata, hogy kezelje a szövegben előforduló sormázó karaktereket (\$ és #).

```

    ha ((c<>'#') vagy (elozo='#')) és ((c<>'$') vagy (elozo='$')) akkor
        Kimenet[sor] := Kimenet[sor]+ c
    elágazás vége

    ha ritkit akkor
        Kimenet[sor] := Kimenet[sor]+ ' '
        karakterHely := karakterHely + 1
    elágazás vége

```

FormaBeolvas eljárás:

konstans maxAlahuzas = 50

változók

szovegF: TextFile
 alahuzasHely : tömb[1..maxAlahuzas] egész számokból
 sor, j, alahuzasSzam, karakterHely, alaHuzIr: egész szám
 ritkit, alahuz, ujsor, vanAlahuzas, elozodollar : logikai
 c, elozo : karakter

Fájlmenyítés (szovegF)

ritkit := hamis; ujsor:=hamis; vanAlahuzas := hamis

alahuzasSzam := 0

alahuzasSzam := 0

elozodollar := hamis

sor:=0

```

    ciklus amíg nem vége(szovegF)
        karakterolvas (szovegF, c)
        karakterHely := karakterHely + 1

```

„formazoKarakterek” rész

```

    ha c = <újsor karakter> akkor
        sor:= sor+1
        ujsor := igaz
        karakterhely := 0

```

különben

```

    ha ujsor = igaz akkor
        ujsor := hamis

```

különben

„karakterSorVegere” rész

elágazás vége

elágazás vége

ha fájlvége(szovegF) akkor sor:=sor+1

„ujsor” rész

elozo := c

ciklus vége

FájlBezár(szovegF)

Eljárás vége

Forráskód:

„Unit 1” unit

```
{ $mode objfpc } { $H+ }  
interface  
  
uses  
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls,  
  Buttons;  
  
type  
  { TtajakUrlap }  
  
  TtajakUrlap = class(TForm)  
    Button1: TButton;  
    szoveg: TMemo;  
    procedure Button1Click(Sender: TObject);  
    procedure FormCreate(Sender: TObject);  
  private  
    { private declarations }  
  public  
    { public declarations }  
  end;  
  
var  
  tajakUrlap: TtajakUrlap;  
  
implementation  
{ TtajakUrlap }  
  
procedure TtajakUrlap.FormCreate(Sender: TObject);  
begin  
  szoveg.Lines.LoadFromFile('tajekoztato.dat');  
end;  
  
procedure TtajakUrlap.Button1Click(Sender: TObject);  
begin  
  Close;  
end;  
  
initialization  
  { $I unit1.lrs }  
end.
```

„Feldolgoz” unit:

unit feldolgoz;

{\$mode objfpc}{\$H+}

interface

uses

Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls,
Menus, Buttons, Unit1;

type

{ TForm1 }

TForm1 = class(TForm)

formazoGomb: TButton;

autoForma: TCheckBox;

MainMenu1: TMainMenu;

Memo1: TMemo;

fajl_menu: TMenuItem;

megnyitas: TMenuItem;

kilep: TMenuItem;

MenuItem1: TMenuItem;

kezikonyv: TMenuItem;

nevjegy: TMenuItem;

OpenDialog1: TOpenDialog;

ment: TMenuItem;

SaveDialog1: TSaveDialog;

procedure FormCreate(Sender: TObject);

procedure Memo1Change(Sender: TObject);

procedure autoFormaChange(Sender: TObject);

procedure formazoGombClick(Sender: TObject);

procedure kezikonyvClick(Sender: TObject);

procedure kilepClick(Sender: TObject);

procedure megnyitasClick(Sender: TObject);

procedure mentClick(Sender: TObject);

procedure nevjegyClick(Sender: TObject);

procedure simaBeolvas;

procedure formaBeolvas;

```
private
  { private declarations }
public
  { public declarations }
end;

var
  Form1: TForm1;

implementation

{ TForm1 }
procedure TForm1.simaBeolvas;
var
  szovegF: TextFile;
begin
  Memo1.Clear;
  AssignFile (szovegF, OpenFileDialog1.FileName);
  Reset (szovegF);
  while not eof(szovegF) do
  begin
  end;
end;

procedure TForm1.formaBeolvas;
const maxAlahuzas = 50;
var
  szovegF                : TextFile;
  alahuzasHely            : array[1..maxAlahuzas] of integer;
  sor, j, alahuzasSzam, karakterHely,
  alaHuzIr               : integer;
  ritkit, alahuz, ujsor, vanAlahuzas,
  elozodollar            : boolean;
  c, elozo               : char;
```



```
begin
  Memo1.Clear;
  AssignFile (szovegF, OpenFileDialog1.FileName);
  Reset (szovegF);
  ritkit := false;
  ujsor:=false;
  vanAlahuzas := false;
  alahuzasSzam := 0;
  alahuzasSzam := 0;
  elozodollar := false;
  sor:=0;
  while not eof(szovegF) do
    begin
      read (szovegF, c);
      karakterHely := karakterHely + 1;
      {$INCLUDE includes/formazoKarakterek.inc}
      if (elozo='$') and (c='$') then
        begin
          alahuzasSzam := alahuzasSzam-1;
        end;

      if c = chr(13) then
        begin
          sor:= sor+1;
          ujsor := true;
          karakterhely := 0;
        end else
        begin
          if ujsor = true then
            begin
              ujsor := false;
            end else begin
              {$INCLUDE includes/karakterSorVegere.inc}
            end;
          end;
        end;

      if eof(szovegF) then sor:=sor+1;

      {$INCLUDE includes/ujsor.inc}
      elozo := c;
    end;
  CloseFile(szovegF);
end;
```

```
procedure TForm1.megnyitasClick(Sender: TObject);
begin
    if openDialog1.Execute then
    begin
        //megnyitas.Enabled:=false;
        // autoForma.Visible:=false;
        ment.Enabled:=true;
        formazoGomb.Enabled := true;
        if (autoForma.Checked) then
        begin
            formazoGomb.Caption := 'Fájl újratöltése formázás nélkül';
            Memo1.Clear;
            formaBeolvas;
        end else
        begin
            formazoGomb.Caption := 'Szöveg formázása';
            Memo1.Clear;
            Kimenet.LoadFromFile(OpenDialog1.FileName);
        end;
    end;
end;

procedure TForm1.mentClick(Sender: TObject);
begin
    if SaveDialog1.Execute then
    begin
        Kimenet.SaveToFile(SaveDialog1.FileName+'.txt');
    end;
end;

procedure TForm1.nevjegyClick(Sender: TObject);
begin
    ShowMessage('Szövegformázó alkalmazás.'+chr(13)+chr(13)+'Készítette: Kiss
Endre Farkas'+chr(13)+chr(13)+'Programozás módszertan 3 beadandó
feladat'+chr(13)+chr(13)+'2005 december');
end;

procedure TForm1.Memo1Change(Sender: TObject);
begin
    if Memo1.Text>' ' then
    begin
        ment.Enabled:=true;
    end;
end;
```

```
procedure TForm1.autoFormaChange(Sender: TObject);
begin

end;

procedure TForm1.formazoGombClick(Sender: TObject);
begin

    if formazoGomb.Caption = 'Szöveg formázása' then
    begin
        formaBeolvas;
        formazoGomb.Caption := 'Fájl újratöltése formázás nélkül';
    end else
    begin
        kimenet.LoadFromFile(OpenDialog1.FileName);
        formazoGomb.Caption := 'Szöveg formázása';
    end;

end;

procedure TForm1.kezikonyvClick(Sender: TObject);
begin
    tajekUrlap.Visible := true;
end;

procedure TForm1.kilepClick(Sender: TObject);
begin
    Close;
end;

initialization
    {$I feldolgoz.lrs}
end.
```

Szerző adatai

Név:	Kiss Endre Farkas
ETR azonosító:	KIELAAT.elte
Gyakorlatvezető:	Horváth László
Telefonszám:	06-30-456-3105
E-mail cím:	kissfarkas@t-online.hu