Pásztor Attila

Algoritmizálás és programozás tankönyv az emeltszintű érettségihez

11. Érettségi feladatok

Ebben a fejezetben az elmúlt 4 érettségi időszak emeltszintű informatika érettségik programozási feladatai közül fogunk néhányat megoldani. Elsősorban a működőképes megoldásra koncentrálunk, nem fordítunk majd különös figyelmes az „eleganciára” és a hatékonyságra. A feladatok megoldását részfeladatonként egyből kódolva fogjuk megoldani.

11.1. Lotto (2005. V.)

Magyarországon 1957 óta lehet ötös lottót játszani. A játék lényege a következő: a lottószelvényeken 90 szám közül 5 számot kell a fogadónak megjelölnie. Ha ezek közül 2 vagy annál több megegyezik a kisorsolt számokkal, akkor nyer. Az évek során egyre többen hódoltak ennek a szerencsejátéknak és a nyeremények is egyre nőttek.

Adottak a lottosz.dat[1] szöveges állományban a 2003. év 51 hetének ötös lottó számai. Az első sorában az első héten húzott számok vannak, szóközzel elválasztva, a második sorban a második hét lottószámai vannak stb.

Például:
37 42 44 61 62
18 42 54 83 89
...
9 20 21 59 68

A lottószámok minden sorban emelkedő számsorrendben szerepelnek.
Az állományból kimaradtak az 52. hét lottószámai.
Ezek a következők voltak: 89 24 34 11 64.

Készítsen programot a következő feladatok megoldására!

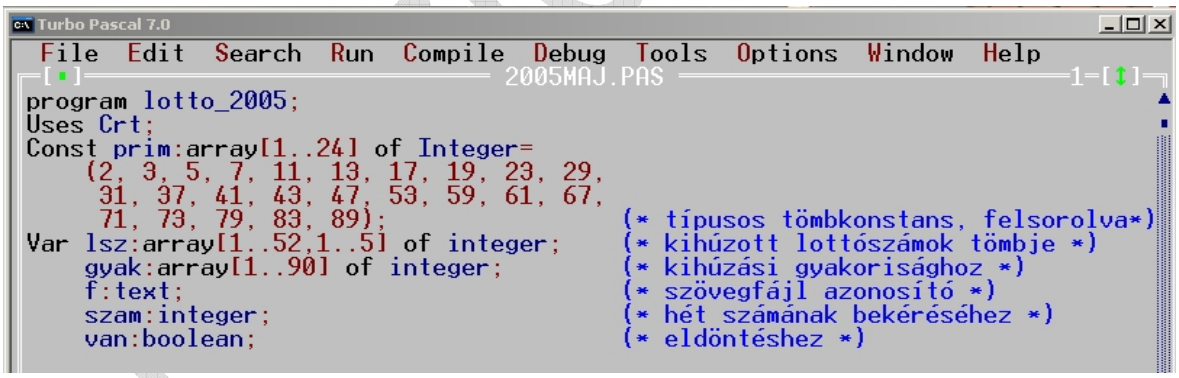
1. Kérje be a felhasználótól az 52. hét megadott lottószámait!
2. A program rendezze a bekért lottószámokat emelkedő sorrendbe! A rendezett számokat írja ki a képernyőre!
3. Kérjen be a felhasználótól egy egész számot 1-51 között! A bekért adatot nem kell ellenőrizni!
4. Írja ki a képernyőre a bekért számnak megfelelő sorszámú hét lottószámait, a lottosz.dat állományban lévő adatok alapján!
5. A lottosz.dat állományból beolvasott adatok alapján döntse el, hogy volt-e olyan szám, amit egyszer sem húztak ki az 51 hét alatt! A döntés eredményét (Van/Nincs) írja ki a képernyőre!
6. A lottosz.dat állományban lévő adatok alapján állapítsa meg, hogy hányszor volt páratlan szám a kihúzott lottószámok között! Az eredményt a képernyőre írja ki!

[1] A lottosz.dat állomány a www.okev.hu címről tölthető le. Az „Érettségi vizsgák” linkre kattintva ki kell választani a „2005. május-júniusi érettségi írásbeli feladatok…” linket, majd az „Emeltszintű írásbeli érettségi vizsgák…” linket, és ezen az oldalon az informatika tárgy mellett letölthetők a források. Az állomány megtalálható a http://pasztora.web.fazekas.hu/tankonyv címen is.

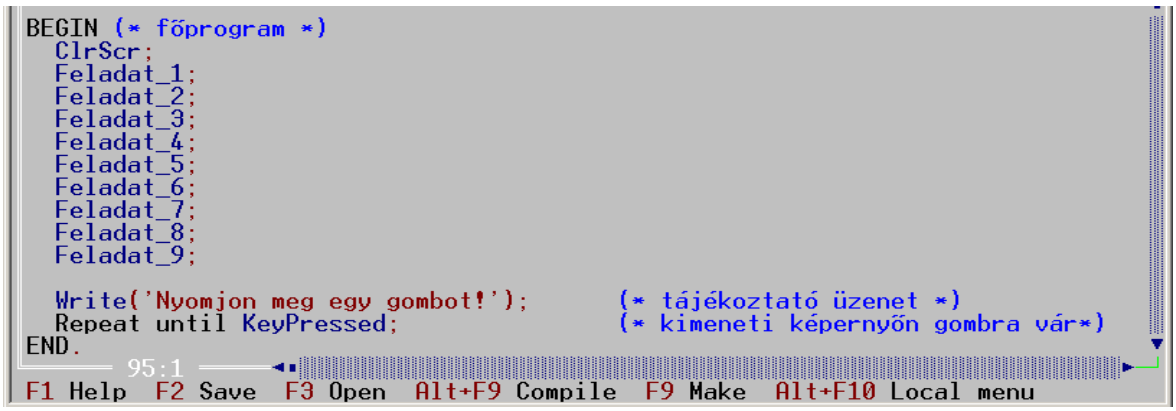
Fűzze hozzá a lottosz.dat állományból beolvasott lottószámok után a felhasználótól bekért, és rendezett 52. hét lottószámait, majd írja ki az összes lottószámot a lotto52.ki szöveges fájlba! A fájlban egy sorba egy hét lottószámai kerüljenek, szóközzel elválasztva egymástól!
Határozza meg a lotto52.ki állomány adatai alapján, hogy az egyes számokat hányszor húzták ki 2003-ban. Az eredményt írja ki a képernyőre a következő formában: az első sor első eleme az a szám legyen ahányszor az egyest kihúzták! Az első sor második eleme az az érték legyen, ahányszor a kettes számot kihúzták stb.! (Annyit biztosan tudunk az értékekről, hogy mindegyikük egyjegyű.)
Példa egy lehetséges eredmény elrendezésére (6 sorban, soronként 15 érték).
4 2 2 4 2 2 6 1 1 2 1 5 2 1 1
1 3 5 0 5 5 2 6 6 5 1 0 6 4 3
3 3 5 4 3 1 4 2 2 4 2 4 1 2 3
4 2 1 2 3 2 2 2 4 4 5 1 3 5 5
5 2 0 2 2 4 4 3 1 3 6 1 5 6 2
4 3 2 2 3 1 1 4 1 3 3 2 1 5 3
Adja meg, hogy az 1-90 közötti prímszámokból melyiket nem húzták ki egyszer sem az elmúlt évben. A feladat megoldása során az itt megadott prímszámokat felhasználhatja vagy előállíthatja! (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89.)

Megoldás:

A program készítés első fázisában az előre láthatóan szükséges változókat deklaráljuk, és elkészítjük a főprogramot. A megadott prímszámokat típusos tömbkonstansban tároljuk (TP megengedi). Az **lsz** tömbbe olvassuk majd be a lottószámokat, és a **gyak** tömbben számolunk előfordulási gyakoriságot.



A főprogram gyakorlatilag a 9 feladat eljárásának meghívását tartalmazza:



Az első eljárás bekér öt számot szóközzel elválasztva (a végén kell csak **ENTER**-t ütni. Semmit nem ellenőriz, de egyből az **lsz** tömb 52. sorában tárolja azokat:

```
Procedure Feladat_1;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  Write('Az 52. hét számai (szóközzel elválasztva): ');
  For i:=1 to 5 do Read(lsz[52,i]);
End;
```

A második feladatban kért rendezést a minimumkiválasztásos rendezéssel valósítjuk meg (az **lsz** tömb 52. sorában dolgozik), majd kiírjuk a rendezett elemeket:

```
Procedure Feladat_2;
Var i,j,MIN:Integer;
    X:Integer;
Begin
  For i:=1 to 4 do                      (* minimum kiválasztásos rendezés *)
    Begin
       MIN:=i;
       For j:=i+1 to 5 do
          If lsz[52,MIN]>lsz[52,j] then MIN:=j;
       X:=lsz[52,i]; lsz[52,i]:=lsz[52,MIN]; lsz[52,MIN]:=X;
    End; (* for i *)
  Write('Az 52. hét számai rendezve: ');  (* kiírás a képernyőre rendezetten*)
  For i:=1 to 5 do Write(lsz[52,i],' ');
  WriteLn;
End;
```

A harmadik feladatban egy számot kell bekérni (1 és 51 között), de nem kell ellenőrizni a beolvasást:

```
Procedure Feladat_3;
Begin
  Write('Adjon meg egy számot 1 és 51 között: ');
  ReadLn(szam);                         (* nem kell ellenőrizni *)
End;
```

A negyedik feladatban ki kell írni a bekért számú hét lottószámait, amihez be kell olvasni a lottosz.dat fájlból az adatokat (ügyeljünk rá, hogy 51 hét adatai vannak benne, az 52. hetet már feltöltöttük):

```
Procedure Feladat_4;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  Assign(f,'lottosz.dat');              (* azonosító hozzárendelés *)
  Reset(f);                             (* megnyitás olvasásra *)
  For i:=1 to 51 do                     (* 51x5 szám beolvasása *)
    ReadLn(f,lsz[i,1],lsz[i,2],lsz[i,3],lsz[i,4],lsz[i,5]);
  Close(f);                             (* fájl bezárása *)
  Write(szam,'. hét lottószámai: ');    (* a megadott heti számok *)
  For i:=1 to 5 do
    Write(lsz[szam,i],' ');
  WriteLn;
End;
```

Az ötödik feladatban el kell dönteni, hogy van-e olyan szám, amit nem húztak ki az állomány 51 heti adatai alapján. Az eldöntés tétel alkalmazását gyakorolhatjuk:

```
Procedure Feladat_5;
Var i,j:Integer;                          (* ciklusváltozók *)
Begin
  For i:=1 to 90 do gyak[i]:=0;           (* gyakorisági tömb előállítása *)
  For i:=1 to 51 do
    For j:=1 to 5 do
      Inc(gyak[lsz[i,j]]);
  i:=1;                                   (* eldöntés tétel *)
  while (i<=90) and (gyak[i]<>0) do i:=i+1;
  van:=(i<=90);
  If van then WriteLn('Van.')             (* kiírás *)
         else WriteLn('Nincs.');
End;
```

A hatodik feladatban megszámolás tételt alkalmazunk és kiírjuk, hogy hány páratlan számot húztak ki (51 hét alatt):

```
Procedure Feladat_6;
Var i,j:Integer;                          (* ciklusváltozók *)
    db:Integer;                           (* hány páratlan szám van *)
Begin
  db:=0;
  For i:=1 to 51 do                       (* csak lottosz.dat adatai kellenek*)
    For j:=1 to 5 do
      If (lsz[i,j] MOD 2 = 1) then db:=db+1;
  WriteLn('Páratlan számok darabszáma: ',db);
End;
```

A hetedik feladatban ki kell írni a teljes 52 heti adatot a lotto52.ki állományba. Vigyázzunk arra, hogy a számok között szóköz legyen, a sorok végén pedig sorvége karakterek!

```
Procedure Feladat_7;
Var i,j:Integer;                           (* ciklusváltozók *)
Begin
  Assign(f,'lotto52.ki');
  ReWrite(f);
  For i:=1 to 52 do                        (* 51x5 szám kiírása *)
    Begin
      For j:=1 to 4 do Write(f,lsz[i,j],' ');
      WriteLn(f,lsz[i,5]);
    End;
  Close(f);                                (* fájl bezárása *)
End;
```

A nyolcadik feladatban ismét elő kell állítani a gyakoriság tömböt, de 52 hétre. A kiszámolt értékeket táblázatosan írjuk ki, a példának megfelelően:

```
Procedure Feladat_8;
Var i,j:Integer;                          (* ciklusváltozók *)
Begin
  For i:=1 to 90 do gyak[i]:=0;           (* gyakorisági tömb előállítása *)
  For i:=1 to 52 do                       (*  most 52 hétre, de nem kell file*)
    For j:=1 to 5 do
      Inc(gyak[lsz[i,j]]);
  WriteLn('Előfordulási statisztika:');   (* tájékoztató szöveg *)
  For i:=1 to 90 do
    Begin
      Write(gyak[i],' ');                 (* 6x15 érték kiírás *)
      If i MOD 15 = 0 then WriteLn;
    End;
End;
```

A kilencedik feladatban a ki nem húzott prímszámokat kell megadni. Igen, ezt is tanultuk: kiválogatás kiírással, aminek a belsejében egy egyszerű eldöntés van:

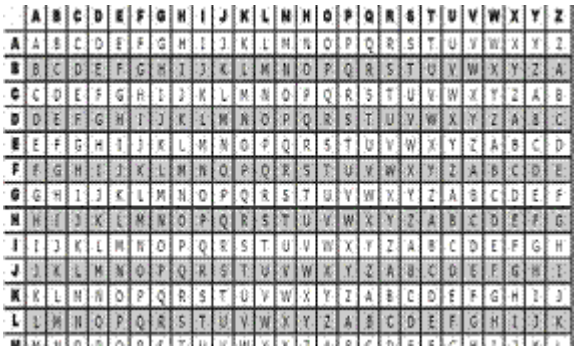
```
Procedure Feladat_9;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  WriteLn('Ki nem húzott prímszámok:');
  For i:=1 to 24 do
    If gyak[prim[i]] = 0 then Write(prim[i],' ');
  WriteLn;
End;
```

Egy lehetséges kimeneti képernyő:

```
Az 52. hét számai (szóközzel elválasztva): 89 24 34 11 64
Az 52. hét számai rendezve: 11 24 34 64 89
Adjon meg egy számot 1 és 51 között: 1
1. hét lottószámai: 37 42 44 61 62
Van.
Páratlan számok darabszáma: 126
Előfordulási statisztika:
4 2 2 4 2 2 6 1 1 2 1 5 2 1 1
1 3 5 0 5 5 2 6 6 5 1 0 6 4 3
3 3 5 4 3 1 4 2 2 4 2 4 1 2 3
4 2 1 2 3 2 2 2 4 4 5 1 3 5 5
5 2 0 2 2 4 4 3 1 3 6 1 5 6 2
4 3 2 2 3 1 1 4 1 3 3 2 1 5 3
Ki nem húzott prímszámok:
19
Nyomjon meg egy gombot!
```

11.2. A Vigenère tábla (2005. X.)

Már a XVI. században komoly titkosítási módszereket találtak ki az üzenetek elrejtésére. A század egyik legjobb kriptográfusának Blaise de Vigenère-nek a módszerét olvashatja a következőkben. A kódoláshoz egy táblázatot és egy ún. kulcsszót használt. A táblázatot (pontosabban egy részletét) a jobb oldali ábra tartalmazza



A tábla adatait a *vtabla.dat* fájlban találja a következő formában.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
FGHIJKLMNOPQRSTUVWXYZABCDE
```

Készítsen programot *kodol* néven a következő feladatok végrehajtására!

1. Kérjen be a felhasználótól egy maximum 255 karakternyi, nem üres szöveget! A továbbiakban ez a nyílt szöveg.
2. Alakítsa át a nyílt szöveget, hogy a későbbi kódolás feltételeinek megfeleljen! A kódolás feltételei:
   - A magyar ékezetes karakterek helyett ékezetmenteseket kell használni. (Például á helyett a; ő helyett o stb.)
   - A nyílt szövegben az átalakítás után csak az angol ábécé betűi szerepelhetnek.
   - A nyílt szöveg az átalakítás után legyen csupa nagybetűs.
3. Írja ki a képernyőre az átalakított nyílt szöveget!
4. Kérjen be a felhasználótól egy maximum 5 karakteres, nem üres kulcsszót! A kulcsszó a kódolás feltételeinek megfelelő legyen! (Sem átalakítás, sem ellenőrzés nem kell!) Alakítsa át a kulcsszót csupa nagybetűssé!
5. A kódolás első lépéseként fűzze össze a kulcsszót egymás után annyiszor, hogy az így kapott karaktersorozat (továbbiakban kulcsszöveg) hossza legyen egyenlő a kódolandó szöveg hosszával! Írja ki a képernyőre az így kapott kulcsszöveget!
6. A kódolás második lépéseként a következőket hajtsa végre! Vegye az átalakított nyílt szöveg első karakterét, és keresse meg a vtabla.dat[2] fájlból beolvasott táblázat első oszlopában! Ezután vegye a kulcsszöveg első karakterét, és keresse meg a táblázat első sorában! Az így kiválasztott sor és oszlop metszéspontjában lévő karakter lesz a kódolt szöveg első karaktere. Ezt ismételje a kódolandó szöveg többi karakterével is!
7. Írja ki a képernyőre és a *kodolt.dat* fájlba a kapott kódolt szöveget!

Példa:

Nyílt szöveg: `Ez a próba szöveg, amit kódolunk!`
Szöveg átalakítása: `EZAPROBASZOVEGAMITKODOLUNK`

Kulcsszó: auto
Kulcsszó nagybetűssé alakítása: AUTO

Nyílt szöveg és kulcsszöveg együtt:

```
E Z A P R O B A S Z O V E G A M I T K O D O L U N K
A U T O A U T O A U T O A U T O A U T O A U T O A U
```

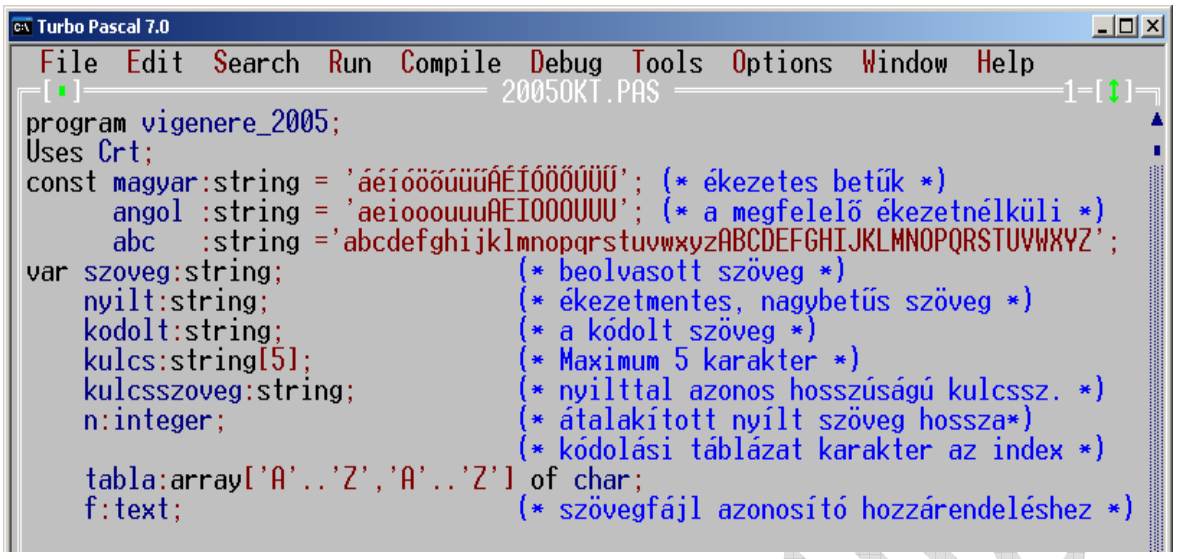
Kódolt szöveg:

```
E T T D R I U O S T H J E A T A I N D C D I E I N E
```

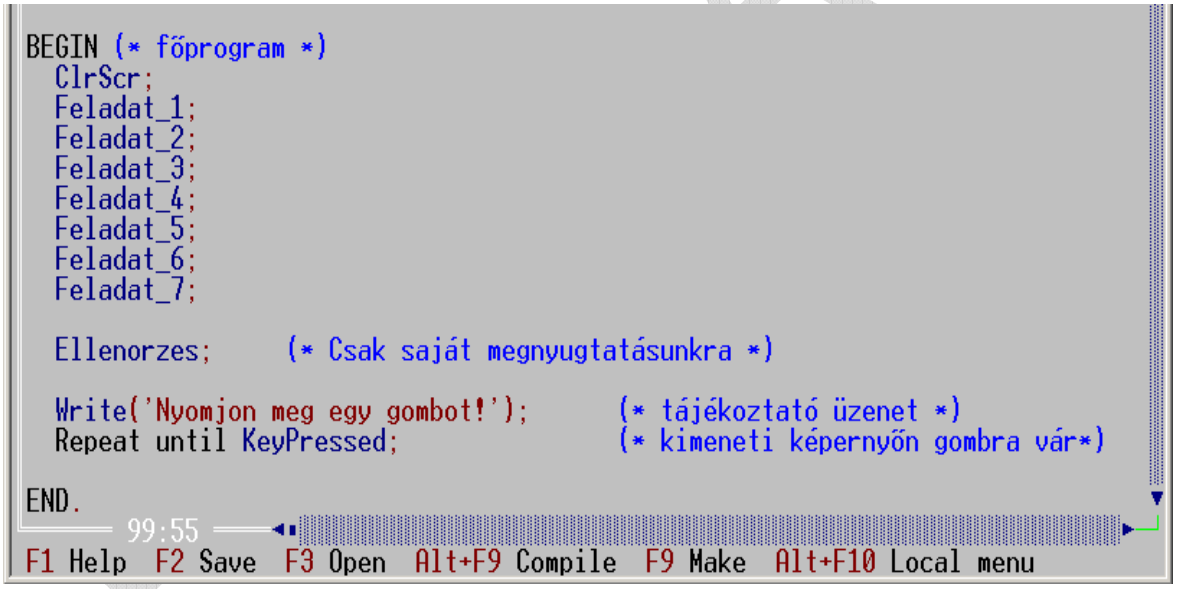
Megoldás:

A program működőképessége érdekében a szükséges változókat deklaráljuk. Az ékezetmentesítéshez szöveges konstansokat fogunk használni: az angol szöveg i-edik karakterében van a magyar karakter helyettesítője.

[2] A vtabla.dat állomány a www.okev.hu címről tölthető le. Az „Érettségi vizsgák” linkre kattintva ki kell választani a „2005. október-novemberi érettségi írásbeli feladatok…” linket, majd az „Emeltszintű írásbeli érettségi vizsgák…” linket, és ezen az oldalon az informatika tárgy mellett letölthetők a források. Az állomány megtalálható a http://pasztora.web.fazekas.hu/tankonyv címen is.



A főprogram a képernyő törlésén kívül meghívja az egyes feladatok eljárásait. Előnyös így elkészíteni a programot, hiszen jól tagolt kódot eredményez. A főprogram végén egy billentyű megnyomására várunk, hogy látható maradjon a kimeneti képernyő. A főprogramban van egy ellenőrző eljárás meghívás is, ez természetesen nem feltétlenül szükséges.



Első lépésként a „nyílt szöveg bekérését, és átalakítását valósítjuk meg, azaz az 1.,2., 3. feladatot.
A szöveg beolvasó eljárásról nincs mit beszélni, íme:

```
Procedure Feladat_1;
Begin
  Write('Kérem a szöveget: ');    (* tájékoztató szöveg *)
  ReadLn(szoveg);                 (* ez a nyílt szöveg *)
End;
```

Az ékezetmentesítés elve az, hogy karakterenként megvizsgáljuk, hogy magyar ékezetes betű-e. Ha az, akkor az angol ábécé megfelelő betűit tartalmazó szöveg ugyanazon sorszá-

mú karakterét tesszük a szövegbe. Ezt követően a szöveget karakterenként az **UpCase** függvénnyel nagybetűsre konvertáljuk. A konvertálást csak a betűkre végezzük el, írásjel esetén továbbmegyünk:

```
Procedure Feladat_2;
Var i:Integer;                    (* ciklusváltozó *)
    p:Integer;                    (* segédváltozó *)
Begin
  nyilt:='';                      (* az átalakított nyílt szöveg még üres *)
  for i:=1 to Length(szoveg) do
    Begin                         (* a szöveg végéig megy karakterenként *)
      p:=Pos(szoveg[i],magyar);   (* a szöveg i-edik karaktere hányadik? *)
      if p<>0 then szoveg[i]:=angol[p];  (* angol betűvel helyettesíti *)
      if Pos(szoveg[i],abc)<>0 then      (* csak a betűt kell átkódolni *)
        nyilt:=nyilt+Upcase(szoveg[i]); (* hozzátesz, nagybetűsen *)
      n:=Length(nyilt);
    End;
End;
```

A 3. feladatban ki kell írni az átalakított nyílt szöveget:

```
Procedure Feladat_3;
Begin
  WriteLn('(3) Az átalakított nyílt szöveg:');
  WriteLn(nyilt);                        (* átalakított nyílt szöveg *)
End;
```

A 4. feladatban a titkosításhoz használt legfeljebb 5 betűs szöveget kell beolvasni (nem kell ellenőrizni, hogy nem üres, csak angol abc-ben szereplő betűket tartalmaz), de nagybetűsre kell alakítani:

```
Procedure Feladat_4;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  Write('Kérem a kulcsszót: ');         (* kérés kiírása *)
  ReadLn(kulcs);                        (* kulcs beolvasás *)
  For i:=1 to Length(kulcs) do          (* nagybetűsre konvertál *)
    kulcs[i]:=UpCase(kulcs[i]);
  WriteLn('Kulcsszó nagybetűssé alakítása: ',kulcs); (* kiírás *)
End;
```

Az 5. feladatban a nyílt szöveggel azonos hosszúságú kulcsszöveget kell készíteni a kulcsszó ismétlésével: annyiszor ismétlem, hogy legalább olyan hosszú legyen, majd levágom a fölösleget. Végül képernyőre írjuk a nyílt szöveget és a kulcsszöveget:

```
Procedure Feladat_5;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  kulcsszoveg:='';
  while Length(kulcsszoveg)<n do        (* üres kulcsszöveggel indulok *)
    kulcsszoveg:=kulcsszoveg+kulcs;     (* ismétlem a kulcsot *)
  kulcsszoveg:=Copy(kulcsszoveg,1,n);   (* pontosan n karakter hosszú lesz *)
  WriteLn('Nyílt szöveg és kulcsszöveg együtt:');
  WriteLn('   ',nyilt);
  WriteLn('   ',kulcsszoveg);
End;
```

A 6. feladatban beolvassuk a vtabla.dat állományból a kódtáblát. Mivel az angol ábécé nagybetűivel indexelt kétdimenziós tömböt választottunk, így a beolvasás nagyon egyszerű. Ezt követi az átkódolás, ami roppant egyszerű: a tábla sorindexe a nyíltszöveg aktuális karaktere lesz, míg oszlopindexe a kulcsszöveg ugyanannyiadik karaktere lesz:

```
Procedure Feladat_6;
Var s,o:Char;                          (* karakterekkel indexelt tömb... *)
    i:Integer;                         (* ciklusváltozó *)
Begin
  Assign(f,'vtabla.dat');              (* kódtábla fájlnév hozzárendelés *)
  Reset(f);                            (* megnyitás olvsásra *)
  for s:='A' to 'Z' do begin           (* ciklus sor és oszlop szerint *)
    for o:='A' to 'Z' do Read(f,tabla[s,o]); (* 26 karaktert beolvas *)
    ReadLn(f);                      (* sorvég karaktereket is olvasni kell *)
  end;
  Close(f);                            (* fájl bezárás *)


  kodolt:='';                          (* kódolt szöveg még üres *)
  for i:=1 to Length(nyilt) do         (* végigmegy a kódolandó szövegen *)
    kodolt:=kodolt+tabla[nyilt[i],kulcsszoveg[i]]; (* átkódolt betű *)
End;
```

A 7. feladatban az átkódolt szöveget ki kell írni a képernyőre és egy kodolt.dat nevű állományba. Ez sem ismeretlen számunkra:

```
Procedure Feladat_7;
Begin
  WriteLn('A kódolt szöveg:');         (* kódolt szöveg képernyőre írása *)
  WriteLn(kodolt);
  Assign(f,'kodolt.dat');              (* kimeneti fájlnév hozzárendelése *)
  ReWrite(f);                          (* megnyitás felülírásra *)
  WriteLn(f,kodolt);                   (* kódolt szöveg fájlba írása *)
  Close(f);                            (* fájl bezárás *)
End;
```

Ezzel a feladatot befejeztük, de a kipróbálás előtt még készítsünk egy ellenőrző eljárást: a kódolt szöveget alakítsuk vissza. Haladva a kódolt szövegben az i-edik karakternek meg kell egyeznie a táblázat kódolandó (ismeretlen) karakternek megfelelő sor kulcsszöveg ugyanannyiadik karakterének megfelelő oszlopában lévő karakterrel. Mivel a tömb tartalom szerint nem indexelhető, ezért ezt egy ciklussal lehet elvégezni.

```
Procedure Ellenorzes;                  (* próbáljuk meg visszaalakítani *)
Var i:Integer;
    s:Char;
    szoveg1:string;                    (* ez lesz a visszaalakított *)
Begin
  szoveg1:='';                         (* kezdetben még üres *)
  for i:=1 to Length(kodolt) do        (* végigmegy a kódolt szövegen *)
    for s:='A' to 'Z' do               (* amikor az s. sorban épp ez van *)
      if kodolt[i]=tabla[s,kulcsszoveg[i]] then     (* a megf. oszlopban ..*)
        szoveg1:=szoveg1+s;
  WriteLn('Visszakódolt szöveg:');
  WriteLn(szoveg1);
  If szoveg1=nyilt then WriteLn('Megegyezik.')
                   else WriteLn('Hibás.');
End;
```

A konkrét feladatra így néz ki a kimeneti képernyő

```
Kérem a szöveget: Ez a próba szöveg, amit kódolunk!
(3) Az átalakított nyílt szöveg:
EZAPROBASZOVEGAMITKODOLUNK
Kérem a kulcsszót: auto
Kulcsszó nagybetűssé alakítása: AUTO
Nyílt szöveg és kulcsszöveg együtt:
  EZAPROBASZOVEGAMITKODOLUNK
  AUTOAUTOAUTOAUTOAUTOAUTOAU
A kódolt szöveg:
ETTDRIUOSTHJEATAINDCDIEINE
Visszakódolt szöveg:
EZAPROBASZOVEGAMITKODOLUNK
Megegyezik.
Nyomjon meg egy gombot!
```

Befejezésül teszteljük a programot más szövegekkel is. Ellenőrizzük, hogy a kodolt.dat állományban is megtalálható-e a kódolt szöveg!

11.3. Fehérje (2006. V.)

A fehérjék óriás molekulák, amelyeknek egy része az élő szervezetekben végbemenő folyamatokat katalizálják. Egy-egy fehérje aminosavak százaiból épül fel, melyek láncszerűen kapcsolódnak egymáshoz. A természetben a fehérjék fajtája több millió. Minden fehérje húszféle aminosav különböző mennyiségű és sorrendű összekapcsolódásával épül fel.
Az alábbi táblázat tartalmazza az aminosavak legfontosabb adatait, a megnevezéseket és az őket alkotó atomok számát (az aminosavak mindegyike tartalmaz szenet, hidrogént, oxigént és nitrogént, néhányban kén is van):

Neve	Rövidítés	Betűjele	C	H	O	N	S
Glicin	Gly	G	2	5	2	1	0
Alanin	Ala	A	3	7	2	1	0
Arginin	Arg	R	6	14	2	4	0
Fenilalanin	Phe	F	9	11	2	1	0
Cisztein	Cys	C	3	7	2	1	1
Triptofán	Trp	W	11	12	2	2	0
Valin	Val	V	5	11	2	1	0
Leucin	Leu	L	6	13	2	1	0
Izoleucin	Ile	I	6	13	2	1	0
Metionin	Met	M	5	11	2	1	1
Prolin	Pro	P	5	9	2	1	0
Szerin	Ser	S	3	7	3	1	0
Treonin	Thr	T	4	9	3	1	0
Aszparagin	Asn	N	4	8	3	2	0
Glutamin	Gln	Q	5	10	3	2	0
Tirozin	Tyr	Y	9	11	3	1	0
Hisztidin	His	H	6	9	2	3	0
Lizin	Lys	K	6	14	2	2	0
Aszparaginsav	Asp	D	4	7	4	1	0
Glutaminsav	Glu	E	5	9	4	1	0

Készítsen programot **feherje** néven, ami megoldja a következő feladatokat! Ügyeljen arra, hogy a program forráskódját a megadott helyre mentse!

1. Töltse be az aminosav.txt[3] fájlból az aminosavak adatait! A fájlban minden adat külön sorban található, a fájl az aminosavak nevét nem tartalmazza. Ha az adatbetöltés nem sikerül, vegye fel a fenti táblázat alapján állandóként az első öt adatsort, és azzal dolgozzon!

   Az első néhány adat:

   Gly
   G
   2
   5
   2
   1
   0
   Ala
   A
   3
   7
   2
   1
   0
   …

2. Határozza meg az aminosavak relatív molekulatömegét, ha a szén atomtömege 12, a hidrogéné 1, az oxigéné 16, a nitrogéné 14 és a kén atomtömege 32! Például a Glicin esetén a relatív molekulatömeg $2 \cdot 12 + 5 \cdot 1 + 2 \cdot 16 + 1 \cdot 14 + 0 \cdot 32 = 75$. A következő feladatok eredményeit írja képernyőre, illetve az **eredmeny.txt** fájlba! A kiírást a feladat sorszámának feltüntetésével kezdje (például: 4. feladat)!
3. Rendezze növekvő sorrendbe az aminosavakat a relatív molekulatömeg szerint! Írja ki a képernyőre és az **eredmeny.txt** fájlba az aminosavak hárombetűs azonosítóját és a molekulatömeget! Az azonosítót és hozzátartozó molekulatömeget egy sorba, szóközzel elválasztva írja ki!
4. A **bsa.txt** a BSA nevű fehérje aminosav sorrendjét tartalmazza – egybetűs jelöléssel. (A fehérjelánc legfeljebb 1000 aminosavat tartalmaz.) Határozza meg a fehérje összegképletét (azaz a C, H, O, N és S számát)! A meghatározásánál vegye figyelembe, hogy az aminosavak összekapcsolódása során minden kapcsolat létrejöttekor egy vízmolekula (H_2O) lép ki! Az összegképletet a képernyőre és az **eredmeny.txt** fájlba az alábbi formában írja ki:
5. Például: C 16321 H 34324 O 4234 N 8210 S 2231
6. (Amennyiben a bsa.txt beolvasása sikertelen, helyette tárolja a G, A, R, F, C betűjeleket tízszer egymás után és a feladatokat erre a „láncra” oldja meg!)
7. A fehérjék szekvencia szerkezetét hasításos eljárással határozzák meg. Egyes enzimek bizonyos aminosavak után kettéhasítják a fehérjemolekulát. Például a Kimotripszin enzim a Tirozin (Y), Fenilalanin (F) és a Triptofán (W) után hasít.

[3] Az aminosav.txt állomány a www.okev.hu címről tölthető le. Az „Érettségi vizsgák” linkre kattintva ki kell választani a „2006. május-júniusi érettségi írásbeli feladatok…” linket, majd az „Emeltszintű írásbeli érettségi vizsgák…” linket, és ezen az oldalon az informatika tárgy mellett letölthetők a források. Az állomány megtalálható a http://pasztora.web.fazekas.hu/tankonyv címen is.

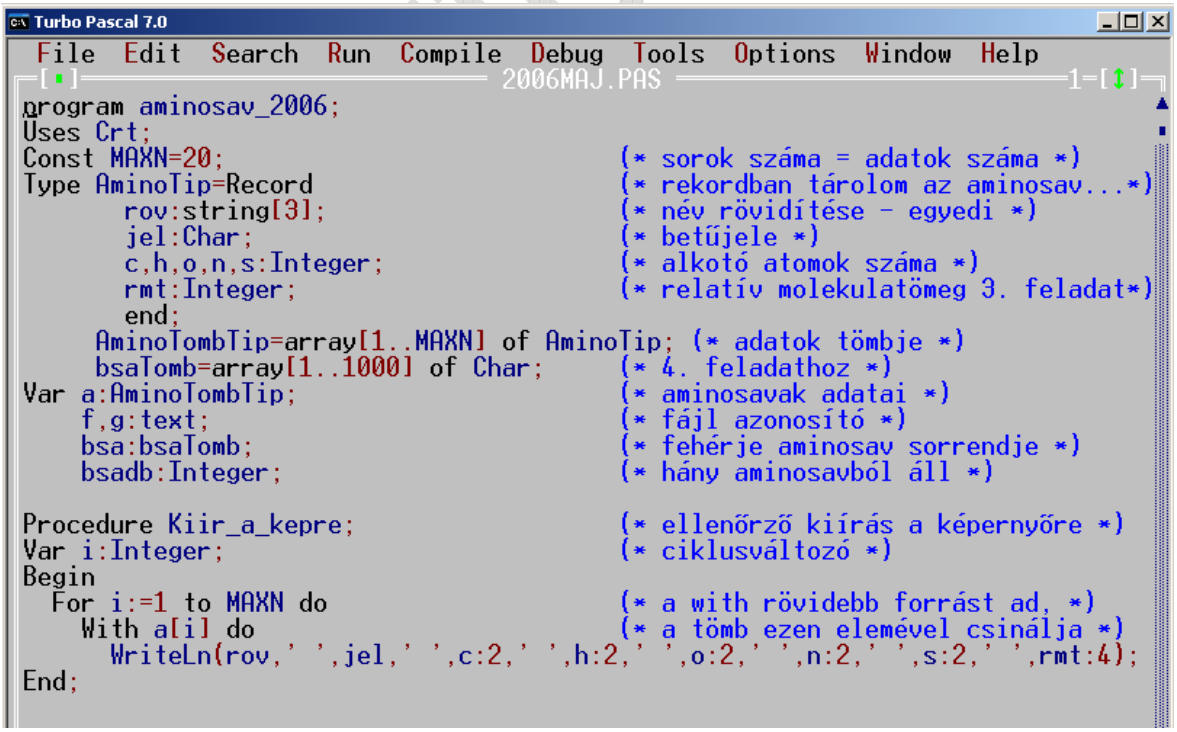
8. Határozza meg, és írja ki képernyőre a Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabjának hosszát és az eredeti láncban elfoglalt helyét (első és utolsó aminosavának sorszámát)! A kiíráskor nevezze meg a kiírt adatot, például: „kezdet helye:”!
9. Egy másik enzim (a Factor XI) az Arginin (R) után hasít, de csak akkor, ha Alinin (A) vagy Valin (V) követi. Határozza meg, hogy a hasítás során keletkező első fehérjelánc részletben hány Cisztein (C) található! A választ teljes mondatba illesztve írja ki a képernyőre!

Megoldás:

Először szeretném megjegyezni, hogy a feladat megoldható rekord adattípus használata nélkül is, de véleményem szerint „elegánsabb” megoldás születhet használatával, és kifejezetten erre való. Felvehettem volna oszloponként egydimenziós tömböket, de például a rendezésnél (3. feladat) sokkal többet kellett volna kódolni.

Először készítsük el a program vázát: konstansként definiálom az aminosav.txt állomány sorainak számát (mivel nincs megadva rá maximum, ezért feltételezem, hogy ennél több nem lehet). Deklarálom egy aminosav jellemzőit (rövidítés, betűjel, és az öt atom darabszáma) az AminoTip típussal. Felveszek egy tömböt, amelyik ilyen rekordból tartalmazhat 20-at. A rekordban található még egy tulajdonság: a relatív molekulatömeg (rmt). Ez a 3. feladathoz hasznos, hiszen ezen tulajdonság (mező) szerint kell majd a rekordokat rendezni. Előkészítem későbbi feladatokat is: felveszem a BSA nevű fehérje aminosav sorrendjének tárolására szolgáló tömböt, és darabszám változóját is.

A deklarációs rész kódja:



A főprogram a szokásos kényelmi funkciókon kívül a feladat sorszámának megfelelő eljárás hívását tartalmazza. A főprogram kódja:

```
BEGIN (* főprogram *)
  ClrScr;
  Feladat_1;
  Feladat_2;
  Feladat_3;
  Feladat_4;
  Feladat_5;
  Feladat_6;

  Write('Nyomjon meg egy gombot!');       (* tájékoztató üzenet *)
  Repeat until KeyPressed;                (* kimeneti képernyőn gombra vár*)
END.
      130:1
F1 Help  F2 Save  F3 Open  Alt+F9 Compile  F9 Make  Alt+F10 Local menu
```

Az első feladatban be kell olvasni az „aminosav.txt” file tartalmát a rekordokat tartalmazó a nevű tömbbe. Az adatok külön sorban vannak, így minden mezőt különálló **ReadLn**-nel lehet beolvasni Érdemes megfontolni, hogy hogyan módosulna a program, ha egy aminosav adatai egy sorban lennének szóközzel, vagy más alkalmas karakterrel elválasztva. A kód:

```
Procedure Feladat_1;
Var i:Integer;
Begin
  Assign(f,'aminosav.txt');               (* fájl azonosító hozzárendelése *)
  Reset(f);                               (* megnyitás olvasásra *)
  for i:=1 to MAXN do                     (* adatok ciklikus beolvasása *)
    Begin
      ReadLn(f,a[i].rov);
      ReadLn(f,a[i].jel); ReadLn(f,a[i].c);
      ReadLn(f,a[i].h); ReadLn(f,a[i].o);
      ReadLn(f,a[i].n); ReadLn(f,a[i].s);
    End;
  Close(f);                               (* állomány bezárás *)
  (* Kiir_a_kepre; *)
End;
```

A beolvasás helyességét mindig érdemes ellenőrizni, ezért készítsünk egy **Kiir_a_kepre** nevű eljárást, mely az a tömb tartalmát (oszlopokba rendezve) kiírja a képernyőre. Ha meggyőződtünk a beolvasás helyességéről, akkor ezt az eljáráshívást megjegyzésbe tehetjük. A teszteléshez készített kiíró eljárás:

```
Procedure Kiir_a_kepre;                   (* ellenőrző kiírás a képernyőre *)
Var i:Integer;                            (* ciklusváltozó *)
Begin
  For i:=1 to MAXN do                     (* a with rövidebb forrást ad, *)
    With a[i] do                          (* a tömb ezen elemével csinálja *)
      WriteLn(rov,' ',jel,' ',c:2,' ',h:2,' ',o:2,' ',n:2,' ',s:2,' ',rmt:4);
End;
```

A második feladatban az a tömb rekordjain haladva ki kell számítani a relatív molekulatömeget (összegezni kell az atomok darabszámával megszorzott relatív atomtömeget). Mivel az atomok relatív atomtömege „ritkán” változik, ezért beépítettem a konkrét értékeket a kódba. Most vesszük a hasznát annak, hogy felvettünk az aminosav rekordjába egy relatív molekulatömeg változót is (**rmt**). Természetesen eltárolhatnánk ezeket az adatokat egy külön tömbben, de a következő feladat sokkal bonyolultabb lenne. Ezt az eljárást is teszteljük a **Kiir_a_kepre** eljárással.. Az utolsó oszlopban jelennek meg a relatív molekulatömeg értékek. Jól látható, hogy a sorok nincsenek rendezve relatív molekulatömeg szerint.

A 2. feladat eljárása:

```
Procedure Feladat_2;
Var i:Integer;                          (* ciklusváltozó *)
Begin
  For i:=1 to MAXN do                   (* végigmegyek a tömbön *)
    Begin                               (* a tömg egy rekordjára ilyen *)
      With a[i] do                      (* módon lehet egyszerűen hivatkozni*)
        rmt:=c*12+h+o*16+n*14+s*32;     (* az összegzés (nem kell a[i],mező)*)
    End;
  (* Kiir_a_kepre; *)
End;
```

A kód tartalmaz egy újdonságot is, a **With a[i] do** utasítást. Az utasítás lehetővé teszi, hogy ne kelljen a mezőkre történő hivatkozásnál a tömb indexét mindig kiírni.

A harmadik feladatban relatív molekulatömeg szerint kell rendezni a beolvasott adatokat. Mivel mi rekordokban tároltuk az aminosavak adatait, agy a rendezés során az adatok együtt mozgathatók. Most is minimumkiválasztásos rendezést használjuk. A rendezés végén a képernyőn teszteljük az eredményt (később megjegyzésbe tesszük), és a képernyő mellett az „eredmeny.txt” állományba is kiírjuk a kért adatokat (rövidítés, relatív molekulatömeg). Javaslom, hogy a kimeneti fájlra más azonosítót használj, mert így elkerülheted, hogy az esetleg nyitva maradt bemenő fájlt írd felül! A kiírás végén bezárjuk az állományt, bár kell majd írni bele még.

```
Procedure Feladat_3;
Var i,j:Integer;
    MIN:Integer;
    X:AminoTip;
Begin
  For i:=1 to MAXN-1 do                 (* minimum kiválasztásos rendezés *)
    Begin
       MIN:=i;
       For j:=i+1 to MAXN do
          If a[MIN].rmt>a[j].rmt then MIN:=j;
       X:=a[i]; a[i]:=a[MIN]; a[MIN]:=X; (* itt látszik a rekord előnye *)
    End; (* for i *)
  (* Kiir_a_kepre; *)
  Assign(g,'eredmeny.txt');             (* fájl azonosító hozzárendelése *)
  ReWrite(g);                           (* megnyitás felülírásra VIGYÁZAT! *)
  WriteLn('3.feladat');                 (* feladat képernyőre *)
  WriteLN(g,'3.feladat');               (* fájlba *)
  For i:=1 to MAXN do                   (* adatok ciklikus beolvasása *)
    Begin
      WriteLn(a[i].rov,' ',a[i].rmt);   (* képernyőre ír *)
      WriteLn(g,a[i].rov,' ',a[i].rmt); (* fájlba ír *)
    End;
  Close(g);                             (* állomány bezárás *)
End;
```

A negyedik feladatban beolvasom a „bsa.txt” állomány tartalmát egy karaktereket tartalmazó tömbbe. Mivel nem ismerem az adatok számát, ezért a fájl végéig olvasok. A fehérje összegképletének meghatározásához végig megyek a bsa tömb elemein, minden karakterhez megkeresem az aminosavak adatait tartalmazó tömb megfelelő rekordját (nem keresés, hanem kiválasztás, mert tudom, hogy van megfelelő), és a rekordban szereplő atom darabszámokat összegzem. Az összegzés végén a kapcsolódások miatti vízvesztéssel korrigálom az eredményt (eggyel kevesebb kapcsolódás van, mint aminosav a tömbben). Az eredményt a képernyőre és az előző „eredmény.txt” állományba is ki kell írni. A kimeneti fájlt úgy kell megnyitni, hogy **felülírás** helyett **hozzáfűzés** történjen. A kód az alábbi:

```
Procedure Feladat_4;
Var i,j:Integer;                          (* ciklusváltozó *)
    ajel:Char;                            (* a bsa tömbben lévő akt. kar. *)
    cdb,hdb,odb,ndb,sdb:Integer;          (* atom darabszámok *)
Begin
  Assign(f,'bsa.txt');                    (* állománynév hozzárendelés *)
  Reset(f);                               (* megnyitás olvasásra *)
  bsadb:=0;                               (* közben meg is számolom *)
  While not eof(f) do                     (* ismeretlen fájlméretnél végéig *)
    Begin
      bsadb:=bsadb+1;
      ReadLn(f,bsa[bsadb]);               (* egysor (karakter) beolvasása *)
    End; (* while ...e of*)
  Close(f);
  {
  For i:=1 to bsadb do Write(bsa[i]); WriteLn; (* ellenőrző kiírás *)
  }
  cdb:=0; hdb:=0; odb:=0; ndb:=0; sdb:=0;(* változók inicializálása *)
  For i:=1 to bsadb do                    (* végigmegyünk bsa tömbön *)
    Begin
      ajel:=bsa[i];                       (* aminosav jele*)
      j:=1;                               (* kiválasztás az a tömbben *)
      While a[j].jel<>ajel do j:=j+1;     (* biztosan van ilyen eleme *)
      cdb:=cdb+a[j].c;                    (* darabszámok aktualizálása *)
      hdb:=hdb+a[j].h;
      odb:=odb+a[j].o;
      ndb:=ndb+a[j].n;
      sdb:=sdb+a[j].s;
    End;
  hdb:=hdb-(bsadb-1)*2;                   (* korrekció vízkilépés miatt *)
  odb:=odb-(bsadb-1);
  Assign(g,'eredmeny.txt');               (* fájl azonosító hozzárendelése *)
  Append(g);                              (* megnyitás hozzáfűzésre VIGYÁZAT!*)
  WriteLn('4.feladat');                   (* kiírás képernyőre *)
  WriteLn(g,'4.feladat');                 (* kiírás fájlba *)
  WriteLn('C ',cdb,' H ',hdb,' O ',odb,' N ',ndb,' S ',sdb);
  WriteLn(g,'C ',cdb,' H ',hdb,' O ',odb,' N ',ndb,' S ',sdb);
  Close(g);                               (* szerintem illik lezárni *)
End;
```

Az 5. feladatban az enzimek hatására megszakadó lánc leghosszabb részét kell meghatározni. Megtehetnénk, hogy kiválogatjuk a részek kezdő és végpontjait, majd ezek közül keresünk maximálist, de a két algoritmus összeépítése is egyszerű:

Először azt feltételezem, hogy az első elem az első rész: ennek megfelelő kezdőponttal, végponttal és hosszal. Ezt követően végig megyek a **bsa** tömb elemein (praktikusan számláló ciklussal), és amennyiben valamelyik (ezért van vagy kapcsolat a feltételek között) megadott aminosav jelét találom, akkor törni fog a lánc. A lánc törésénél megvizsgálom, hogy a most kialakult szakasz hosszabb-e, mint az eddigi leghosszabb. Ha hosszabb, akkor eme új szakasz adatait tárolom el leghosszabbként. Ha hosszabb, ha nem, akkor is a következő aminosavnál kezdődik az új szakasz.

A szakasz hosszának számítása felfogás kérdése. Példa nem fogalmaz egyértelműen, hogy hossz alatt a szakaszt alkotó aminosavak darabszámát érti, vagy az aminosav-aminosav szakaszok számát (ezt adja a végpont-kezdőpont érték). A feladat hivatalos megoldásából látszik, hogy az első választ várták, ezért az általam kiszámított hossz értéket eggyel meg kell növelni.

Az eredményt most csak a képernyőre kell kiírni, de ügyelni kell arra, hogy a kért szöveges információ is benne legyen!

```
Procedure Feladat_5;
Var i,j:Integer;
    le,lu,l:Integer;
Begin
  le:=1;                                (* leghosszabb szakasz eleje *)
  lu:=1;                                (* legosszab szakasz vége *)
  l:=lu-le;                             (* hossz - pontosabban hossz-1 *)
  j:=1;                                 (* aktuális szakasz kezdete *)
  For i:=1 to bsadb do                  (* végigmegy a bsa tömbön *)
    Begin
      If Pos(bsa[i],'YFW')>0 then       (* törik *)
        Begin
          If i-j>l then                 (* hosszabb, mint az eddigi max *)
            Begin
              lu:=i;le:=j;l:=lu-le;     (* új leghosszabb szakasz értékei *)
            End;
          j:=i+1;                       (* mindeképpen új szakasz kezdődik *)
        End;
    End;
  l:=l+1;                               (* a különbség+1 adja a hosszt *)
  WriteLn('5.feladat');                 (* kiírás a képernyőre *)
  WriteLn('Kezdete: ',le,' vége: ',lu,' hossza: ',l);
End;
```

A 6. feladatban egy másik enzim által szétszakított lánc első részében kell megszámolni egy megadott aminosav számát. A feladat elvégezhető lenne úgy, hogy **kiválogat**juk az első szakasz aminosavait, majd **megszámol**juk a benne lévő adott tulajdonságú aminosavakat.

Az elemi algoritmusok összeépítése itt is hatékonyabb megoldást ad: Addig összegezzük a **bsa** tömbben található megfelelő (cisztein) aminosavak darabszámát, amíg a fehérje el nem törik az enzim hatására. A törés feltétele most kicsit bonyolultabb, az **és** és a **vagy** kapcsolatokat helyesen kell zárójelezni!

A feladat szövegéből biztosak lehetünk abban, hogy van ilyen törés, tehát a tömbön haladva nem kell figyelnünk arra, hogy biztosan benne maradjuk. A végén kiíratjuk az eredményt (a kért formában) a képernyőre. Ha ez nem így lenne, akkor még egy feltételt be kellene írni.

Eddig általában számláló, vagy elöltesztelő ciklust használtunk, most hátultesztelő ciklust készítünk. Ügyelni kell arra, hogy a kódban a kilépési feltételt kell megadni, és nem a bennmaradásit!

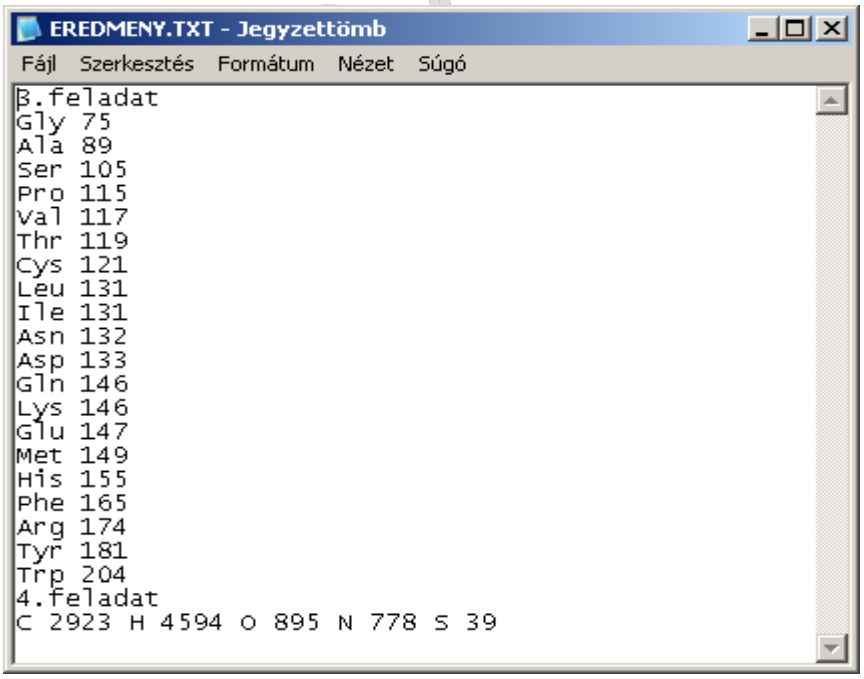
```
Procedure Feladat_6;
Var i,j:Integer;
    db:Integer;
Begin
  db:=0;
  i:=0;
  Repeat
    i:=i+1;
    If bsa[i]='C' then db:=db+1;
  until (bsa[i]='R') and ((bsa[i+1]='A') or (bsa[i+2]='V'));

  WriteLn('6.feladat');
  WriteLn('Az első fehérjeláncban ',db,' cisztein van.');
End;
```

A teszteléshez használt kiírások kitörölhetők, de megjegyzésként benne maradhatnak a forrásban. Érdemes még egyszer megtekinteni, hogy mind a képernyőn, mind a kimeneti fájlban a feladatban elvárt tartalom olvasható. Íme a képernyő tartalma:

```
3.feladat
Gly 75
Ala 89
Ser 105
Pro 115
Val 117
Thr 119
Cys 121
Leu 131
Ile 131
Asn 132
Asp 133
Gln 146
Lys 146
Glu 147
Met 149
His 155
Phe 165
Arg 174
Tyr 181
Trp 204
4.feladat
C 2923 H 4594 O 895 N 778 S 39
5.feladat
Kezdete: 261 vége: 306 hossza: 46
6.feladat
Az első fehérjeláncban 12 cisztein van.
Nyomjon meg egy gombot!
```

Az „eredmeny.txt” kimeneti fájl tartalma:



Mindkét tartalom megfelel a feladat szövegének. Befejeztük a megoldást.