# Az első program

## Tartalom


1. Első lépés ..... 2
   1.1. A feladat ..... 2
   1.2. Specifikáció ..... 2
   1.3. Algoritmus ..... 2
   1.4. Kód ..... 3
2. Második lépés ..... 5
   2.1. A feladat ..... 5
   2.1. Specifikáció ..... 5
   2.2. Algoritmus ..... 5
   2.3. Kód ..... 6
3. Harmadik lépés ..... 6
   3.1. A feladat ..... 7
   3.2. Specifikáció ..... 7
   3.3. Algoritmus ..... 7
   3.4. Kód ..... 7
4. Negyedik lépés ..... 8
   4.1. A feladat ..... 8
   4.2. Specifikáció ..... 8
   4.3. Algoritmus ..... 8
   4.4. Kód ..... 9
5. Ötödik lépés ..... 11
   5.1. A feladat ..... 11
   5.1. Specifikáció ..... 11
   5.2. Algoritmus ..... 11
   5.3. Kód ..... 11
6. Hatodik lépés ..... 12
   6.1. A feladat ..... 12
   6.1. Specifikáció ..... 12
   6.2. Algoritmus ..... 12
   6.3. Kód ..... 12

Figyeljük meg, miként készül el az informális feladatszövegből először a **formális leírás**a, majd a megoldó program **algoritmus**a és (Pascal) **kód**ja! Vegyük észre, hogy az egyes lépésekben felhasználjuk az előző lépés(ek)ben már rögzített döntéseinket!

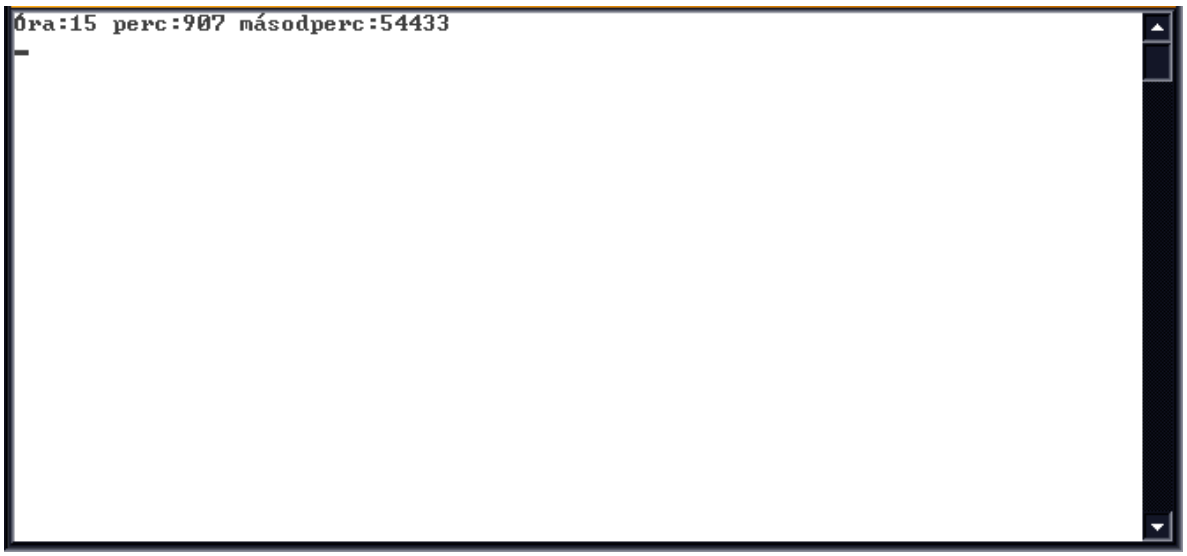
Több lépésben oldunk meg néhány, rokon feladatot, hogy könnyen legyen felfedezhető minden algoritmikus és Pascal nyelvi elem tudnivalói, és mindez viszonylag kevés gépelés árán.

# 1. Első lépés

## 1.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől?

Valami efélére gondolunk:




**1. ábra:** Egy screen-shot az ElsoPrgA futásáról 15:07 környékén.
Mi látszik azonnal? A program egy konzol- vagy más szóval parancsablakos alkalmazás.

## 1.2. Specifikáció

Négy mondanivalónk van: a **bemenet** és a **kimenet** adatainak leírása (névvel és alaphalmazzal ellátása), a bemenettel, illetve a kimenettel szembeni **elvárás**aink formalizálása. Használunk majd néhány halmazt ($\mathbb{N}$: természetes számok, $\mathbb{L}$: logikai értékek, $\mathbb{S}$: szövegek), egy halmaz- ($\in$) és néhány logikai műveletet ($\wedge,\vee,\neg$).

Be: Ó,P,M$\in\mathbb{N}$ *[a pillanatnyi idő óra:perc:másodperc alakban, amelyet az operációs rendszer szolgáltat]*

Ki: ÉÓ,ÉP,ÉM$\in\mathbb{N}$ *[az éjféltől eltelt órák, percek és másodpercek száma]*

Ef: –

Uf: ÉO=Ó $\wedge$ ÉP=60*Ó+P $\wedge$ ÉM=60*(60*Ó+P)+M

## 1.3. Algoritmus

A specifikációban szereplő adatok a program ún. **globális adat**ait alkotják (amely a programban bárhonnan hozzáférhető). Ezek deklarációjával kezdődik a program. Az eredményadatokat mindjárt megjelenítjük, így megspórolhatók a kimeneti változók.

Az algoritmizáláshoz figyelembe vesszük, hogy létezik –a programozási nyelvekben is!– olyan előredefiniált eljárás (`Idő`), amely képes a program számára hozzáférhetővé tenni a szá-

mítógép belső óráját. Valójában tehát a specifikációban szereplő „Be:” rész nem egy hagyományos beolvasást definiál (ennyiben nem tipikus; cserében viszont egyszerűsíti a dolgunkat).

```
Program ElsőProgram_A:                          ← A program fejsora.
  [a globális adatok:]                          ← Megjegyzés sor „[” és „]” a között.
  Változó                                       ← Adatok leírása (deklarálása).
    Ó,P,M:Egész
[   ÉÓ,ÉP,ÉM:Egész]                             ← Megjegyzés sorrá téve a kimeneti adatokat.
  [a főprogram:]                                ← Megjegyzés sor.
  Idő(Ó,P,M) [lekérdezzük a pillanatnyi időt]   ← Az Idő eljárás hívása (Ó,P,M értéket kap).
  Ki:Ó,60*Ó+P,60*(60*Ó+P)+M                     ← Kiírás a képernyőre.
Program vége.                                   ←A program lezárása.
```

Vegyük észre a program kijjebb-beljebb kezdésekkel történő tagolását! Elv: az összetett szerkezetek alárendeltjeit (törzsét) beljebb kezdjük, miközben a hozzá képest mellérendelt szerkezeteket vele azonos bekezdéssel írjuk.

## 1.4. Kód

Lássuk a Pascal kódot! Tudnunk kell, hogy számos, a Pascal standardon[1] túli bővítést is tartalmaznak az egyes Pascal implementációk, amelyeket a fordító program csak külön „kérésre” fordít a kódhoz. A pillanatnyi időt megadni képes eljárás (`GetTime`) is ilyen „kuriózum”, amelynek van egy 4., számunkra fölösleges paramétere: a századmásodperc. Ezt a többlet adatot is deklarálnunk kell az adatleírásban. Tudnunk kell, hogy a Pascalban számos típus van, amely egészek ábrázolására alkalmas: `Byte` (1 bájtos előjel nélküli), `Integer` (2- vagy 4-bájtos előjeles egész), `Word` (2-bájtos előjel nélküli, azaz nem negatív egész), `LongInt` (4-bájtos előjeles egész)… A `GetTime` azonban `Word`-öt vár el, tehát ezt kell választanunk.

Vegyük észre, hogy mennyire „mechanikus” az algoritmus átültetése a Pascal nyelvre. Az adatok kódolására is ügyeljünk: pl. nem lehetnek ékezetes betűk az azonosítókban!

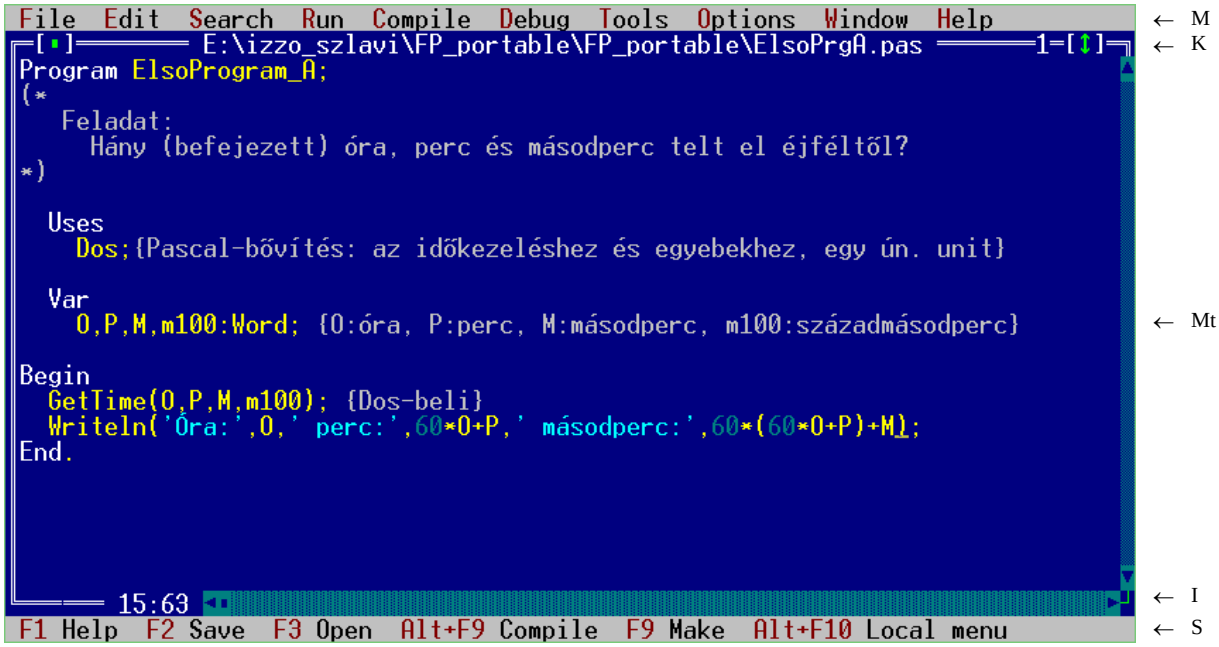
```
Program ElsoProgram_A;                          ← A program fejsora.
(*
   Feladat:
     Hány (befejezett) óra, perc és             ← Megjegyzés sorok ( a „(*” és „*)” közötti rész.
     másodperc telt el éjféltől?
*)
  Uses                                          ← A nem standard könyvtári rutinok fájljainak megadása.
    Dos; {Pascal-bővítés: unit}                 ← Megjegyzéssel követve a „{” és „}” közt.
  Var                                           ← Adatok leírása: O:óra, P:perc,
    O,P,M,m100:Word; {O=óra, …}                 ← M:másodperc,m100:századmásodperc.
                                                  A Word egy előjel nélküli 2-bájtos egész típus.
Begin                                           ← A főprogram tevékenységrészének kezdete.
  GetTime(O,P,M,m100); {Dos-beli}               ← A GetTime eljárás hívása (Ó,P,M,MP értéket kap).
  Writeln('Óra:',O,' perc:',60*O+P,             ← Kiírás a képernyőre, –természetesen– kisérő szöveggel kiegé-
          ' másodperc:',60*(60*O+P)+M);           szítve.
End.                                            ← A program lezárása.
```

Billentyűzzük ezt be a Free Pascal (FP) „hagyományos” (konzolos vagy másként mondva: parancsablakos) fejlesztői környezetében! [2. ábra]

Kezdjük azzal –miután az FP-t elindítottuk–, hogy kijelöljük azt a könyvtárat, amelybe szeretnénk munkánk eredményét, a forráskódot és a futó kódot találni. Használjuk eh-

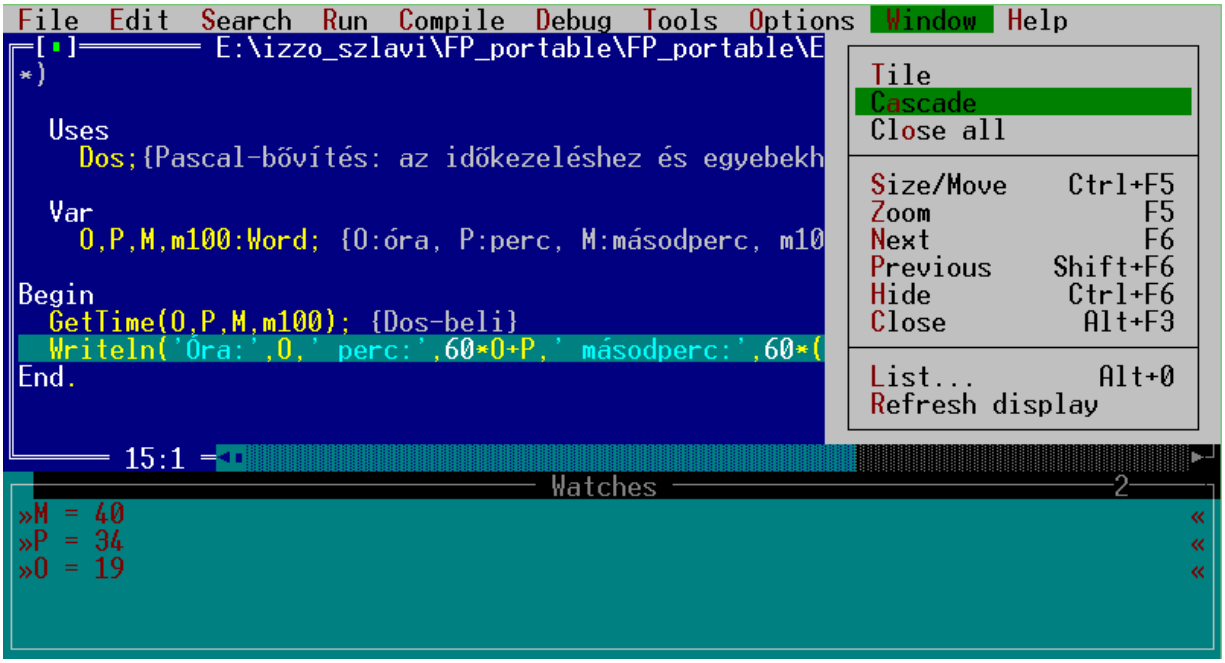
[1] A Pascal standard a Wirth által magalapozott nyelvet jelenti.

hez a „Change dir…” menüt a File menücsoportból (forró billentyűkkel [Alt+F]/[C])! Figyeljünk a forrás gépelése közben a nyelvi elemek „színe változására”, valamint a tagolásra (kijjebb-beljebb kezdésre)!




**2. ábra:** A Free Pascal IDE [2] (Integrált Fejlesztői Környezet).
M: menüsor; Mt: munkaterület; K: a keret azonosítása (a sorszáma, és az éppen benne szerkesztett fájl neve);
I: a Mt-nek az aktuális sor-/oszlop-indexét tartalmazó alsó keretdarabja; S: a forró billentyűk súgósora.

Ha elkészültünk a begépeléssel, fordítsuk le ([Alt+C]/[C] vagy még egyszerűbben: [F9])!
Ha program szintaktikusan helyes, nekiláthatunk a futtatásnak, és megfigyelhetjük a futás „mélyebb rejtelmeit”. Ez utóbbira nem minden fejlesztői környezet képes. Így az alábbi pár gondolat –pontosan így– csak a fentebb említett környezetre vonatkozik.




**3. ábra:** Egy pillanatfelvétel, amely a Free Pascal nyomkövető rendszerének használata közben készült.

[2] http://en.wikipedia.org/wiki/Integrated_development_environment

A futás elemzését teszi lehetővé az IDE nyomkövető rendszere (debugger). Elindítása, illetőleg elindítás után a folytatása, a `Run` menücsoport „`Step over`” pontjával (vagy az `[F8]` forró billentyűvel) lehetséges.[3] Hogy a program adatokra vonatkozó állapotváltozásait is lássuk, nyissuk meg az adatfigyelő ablakot (`Watches`) a `Debug` menücsoport `Watches` pontjával, majd az `[insert]` billentyű többszörös használatával vegyük föl az `O`, a `P` és az `M` változókat! Hogy a továbbiakban mind a két panel (az IDE szóhasználata szerint „ablak”) látható maradjon –s ne kelljen– lapozgatni az `[F6]`-tal, válasszuk a `Window` menücsoport `Cascade` pontját (`[Alt+W]/ [A]`). [3. ábra]

Ha jól kiveséztük programunkat, futtassuk le ne csak az FP környezetben, hanem a fordítás során a háttértáron keletkezett alkalmazást (`ElsoPrgA.exe`) is. Mit tapasztaltunk? [4] Mi lehet a magyarázat? [5] Többek közt e tapasztalatunk motiválja a következő feladatvariánst.

# 2. Második lépés

## 2.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől? De tegyük hozzá, hogy a program kicsit ergonómikusabb és –önálló alkalmazásként is– használható legyen, azaz takarítsa le a képernyőt, mielőtt ráírna, és a befejezés előtt várakozzon felhasználói beavatkozásra!

## 2.1. Specifikáció

Nincs a feladat lényegi részén változás, ezért a specifikáció ugyanaz, mint előbb.

## 2.2. Algoritmus

Az „emberibbé” tétellel nem szokás az algoritmizálás során törődni, mivel ennek megfogalmazása nagyon környezetfüggő szolgáltatásokon alapulnak, és egyébként sem jelentenek –általában– algoritmikus nehézséget.

Annyit azért érdemes –a feladat megváltozásától függetlenül– észrevenni, hogy a kiírásban két számítást megismételve végeztetünk el, amely így fölöslegesen pazarolja a számítógép idejét. Ennek elkerülésére használjuk föl a meglévő (eddig csak a „bemeneti” adatok értékeit tartalmazó) változóinkat, amelyeket módosítunk a célnak megfelelően! Így:

```
Program ElsőProgram_B:
  [a globális adatok:]
  Változó
    Ó,P,M:Egész
  [a főprogram:]
  Idő(Ó,P,M) [lekérdezzük a pillanatnyi időt]
  P:=60*Ó+P [az eltelt idő percekben]          ←értékadás: az éjfél óta eltelt idő, percekben
  M:=60*P+M [az eltelt idő másodpercekben]    ←értékadás: az éjfél óta eltelt idő, másodpercekben
  Ki:Ó,P,M
Program vége.
```

[3] Vegyük észre, hogy FP-ben a futás elindulása után megnyílik egy újabb ablak (a Windows szóhasználata szerint!), így ahhoz hogy az IDE-vel kommunikálhassunk, át kell az IDE ablakába lépnünk!

[4] Az elindulás után a képernyőn egy felvillanás, majd semmi.

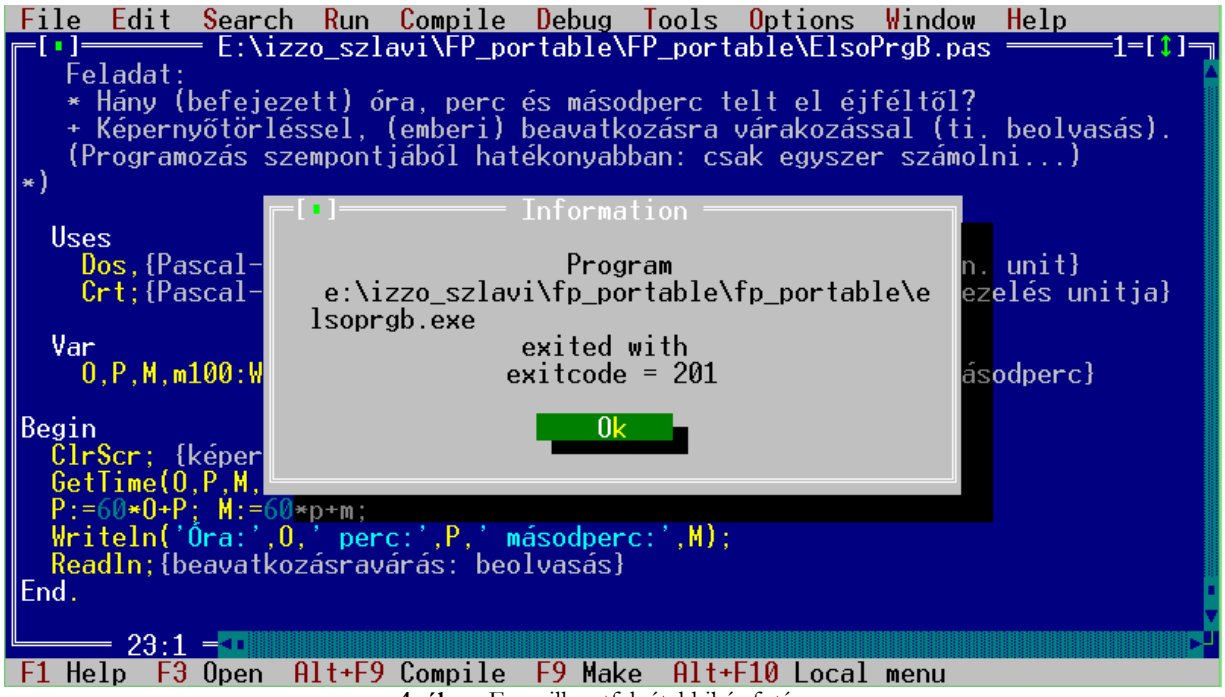
[5] A grafikus operációs rendszerekben (pl. a Windows-ban) minden alkalmazás kap egy (fő) ablakot, amelyben történik a kommunikáció (input/output). Amint az alkalmazás megáll, a hozzárendelt ablak(ok) automatikusan becsukódnak.

## 2.3. Kód

Az ergonómia növelését a képernyő és a billentyűkezelés igényesebb módjával valósítjuk meg. Az ehhez tartozó rutinok egy másik, egy `Crt` nevű unitban találhatók.

```
Program ElsoProgram_B;
  Uses
    Dos,Crt;                    ←  Crt: Pascal-bővítés; az igényesebb képernyő- és billentyű-
                                   kezelés unitja. [6]
  Var
    O,P,M,m100:Word;
Begin
  ClrScr;                       ←  Képernyőtörlés (egy Crt unitbeli rutin), Clear Screen.
  GetTime(O,P,M,m100);
  P:=60*O+P; M:=60*P+M;         ←  Az összetett eredmények kiszámítása előre.
  Writeln('Óra:',O,' perc:',P,
          ' másodperc:',M);
  Readln;                       ←  Beavatkozásra várás: beolvasás (sehova!).
End.
```

Váratlan csalódás ér minket: amint ezt kipróbáljuk, este felé (18 óra után) tapasztaljuk, hogy a program goromba hibajelzés közepette elszáll:




**4. ábra:** Egy pillanatfelvétel hibás futásra.

A bajt az okozza, hogy a választott egész-típus, a `Word` a maga 2 bájtjával kicsi, „csak” 65 537-ig „jó”. (18 óra 13 perc időpont percekben mérve már nagyobb a megengedettnél: 65 580.)

Megoldásként visszatérünk az eredeti specifikációhoz, amelyben még külön létezett kimeneti paraméterekként: `ÉÓ`, `ÉP`, `ÉM`, és ezeket akkorára méretezzük, amekkorára kell. A `LongInt` 4-bájtos, így akár néhány milliárdnyi értéket is képes tárolni egy ilyen típusú változó.

```
Program ElsoProgram_B;
  Uses
    Dos,Crt;
  Var
    O,P,M,m100:Word;
```

[6] Crt ← Cathode Ray Tube (=katódsugárcső): az egykori monitor legjellegzetesebb megjelenítő „alkatrésze”.

```
    EO,EP:Word;
    EM:LongInt;                          ←  A megfelelő méretűre deklarált kimeneti változók.
Begin
  ClrScr;
  GetTime(O,P,M,m100);
  EO:=O; EP:=60*EO+P; EM:=60*EP+M;       ←  Az összetett eredmények kiszámítása.
  Writeln('Óra:',EO,' perc:',EP,
          ' másodperc:',EM);             ←  A kimenőváltozók kiírása.
  Readln;
End.
```

Módosítsa az első változat algoritmusát és kódját, majd próbálja ki ezt a változatot is! A program várakozásánál mit tapasztal?[7]

Megjegyzés: egy másik módosítási javaslat lehet a felvetődött futási probléma kiküszöbölésére, hogy a bemeneti változókat eleve `LongInt`-ként deklaráljuk. Próbálja ki! Mit tapasztal? (Válasz: a `GetTime` eljárás hívásánál jelez hibát a fordító, mondván: nem megfelelő az átadott paraméterek típusa. Persze, mivel az `Word`-öt várt el.)

# 3. Harmadik lépés

## 3.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől? Tovább növeljük a program ergonómiáját azzal, hogy a kiírást egy címmel kezdje, amelyet a legfelső sorában középre igazítva teszi! És a várakozás ügyesebb megszervezése: egyetlen, tetszőleges billentyű lenyomására várjon, és ne echózza azt.

## 3.2. Specifikáció

Nincs a feladat lényegi részén változás, ezért a specifikáció ugyanaz, mint előbb.

## 3.3. Algoritmus

Lényegileg nincs változás.

## 3.4. Kód

Az egyetlen kódolásra maradt meggondolni való: a cím középre igazítása. Ehhez tudnunk kell a képernyő szélességét (80 karakternyi), a cím hosszát (ami lekérdezhető a `Length` függvénnyel), és a `Write` utasításban alkalmazható mező-beállítási lehetőségről.

```
Program ElsoProgram_C;
  Uses
    Dos,Crt;
  Var
    O,P,M,m100:Word;
    EO,EP:Word;
    EM:LongInt;
  Const
    cim:String='A ma eltelt idő';      ←  A cím konstans megadása.
                                          Így is jó lenne: cim='A ma eltelt idő';
Begin
  ClrScr;
  Writeln(cim:40+(Length(cim) Div 2));  ←  A cím mögötti kifejezés azt határozza meg, hogy hol legyen a szöveg vége.
  Writeln;                              ←  Egy üres sort hagyunk (írunk) ki a cím alatt.
  GetTime(O,P,M,m100);
  EO:=O; EP:=60*EO+P; EM:=60*EP+M;
```

[7] A program az esti órákban is sikeresen lefut, továbbá a várakozás közben bármilyen szöveg begépelését megengedi. Valójában csak az `ENTER` billentyűre vár.

```
   Writeln('Óra:',EO,' perc:',EP,
           ' másodperc:',EM);
   ReadKey;                                  ←  Beavatkozásra várás: most billentyű-lenyomásra várakozás
                                                (a ReadKey szintén egy Crt-beli rutin).
End.
```

Az előző változat kódját módosítsa, és próbálja ki!

# 4. Negyedik lépés

## 4.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől? Adjuk meg a napi dátumot, és mondjuk meg, milyen nap van ma! Persze ezeket is igényesen!

## 4.2. Specifikáció

Újdonság most: az utófeltételben az implikáció, a bemeneten és a kimeneten egyaránt megjelenő adat, és az 𝕊-sel jelölt szövegek halmaza (típusa).

Be: Ó,P,M∈ℕ *[a pillanatnyi idő óra:perc:másodperc alakban, amelyet az operációs rendszer szolgáltat]*
É,H,N,HN∈ℕ *[a mai dátum év.hó.nap.hét napja alakban, amelyet az operációs rendszer szolgáltat]*

Ki: ÉÓ,ÉP,ÉM∈ℕ *[az éjféltől eltelt órák, percek és másodpercek száma]*
É,H,N∈ℕ *[a dátum]*
NAP∈𝕊 *[a hét napjának a neve]*

Ef: –

Uf: ÉO=Ó ∧ ÉP=60*Ó+P ∧ ÉM=60*(60*Ó+P)+M ∧ É'=É ∧ H'=H ∧ N'=N ∧
HN=0 → NAP='Vasárnap' ∧ HN=1 → NAP='Hétfő' ∧ … ∧ HN=6 → NAP='Szombat'

Vegye észre az utófeltételben az aposztrofált adatokat! Értelmezés: pl. É'=É azt jelenti, hogy az É feladatparaméter kimenetkori értéke (É') megegyezik a bemenetkorival (É).

## 4.3. Algoritmus

A megvalósításban egy újfajta utasítást, egy feltételes utasítást (az elágazások egyik fajtáját) kell láncba szednünk, ahogy ezt már a specifikáció is sejteti. Természetesen a dátumot lekérdező Dátum eljárás is nóvumként bukkan föl. A Dátum eljárás utolsó paramétereként megadja a nap héten belüli sorszámát: 0 a vasárnaphoz, 1 a hétfőhöz … tartozik.

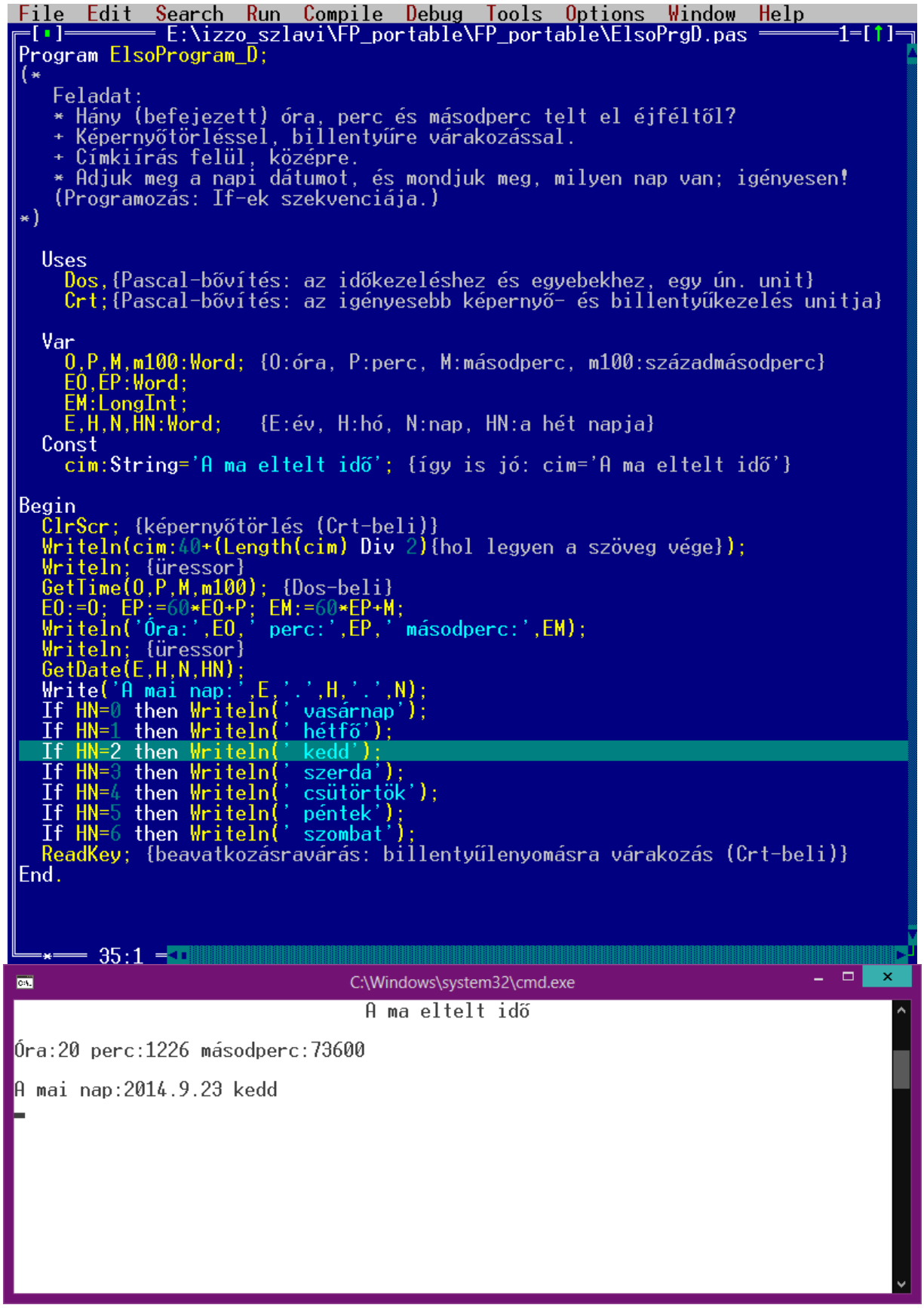
```
Program ElsőProgram_D:
  Változó
    Ó,P,M:Egész
    ÉÓ,ÉP,ÉM:Egész
    É,H,N,HN:Egész                           ←  É:év, H:hó, N:nap, HN:a hét napja
  Konstans
    cim:Szöveg='A ma eltelt idő'
  Idő(Ó,P,M)
  ÉÓ:=Ó; ÉP:=60*ÉÓ+P; ÉM:=60*ÉP+M
  Ki:ÉÓ,ÉP,ÉM
  Dátum(É,H,N,HN)                            ←  A mai dátum „bekérése".
  Ki:É,H,N [nincs soremelés]                 ←  A dátum változatlan kiírása.
  Ha HN=0 akkor Ki:' vasárnap'
  Ha HN=1 akkor Ki:' hétfő'
  Ha HN=2 akkor Ki:' kedd'
  Ha HN=3 akkor Ki:' szerda'                 ←  A HN függvényében a megfelelő nap kiírása, folytatólagosan.
  Ha HN=4 akkor Ki:' csütörtök'
  Ha HN=5 akkor Ki:' péntek'
  Ha HN=6 akkor Ki:' szombat'
Program vége.
```

## 4.4. Kód

A Dátum eljárás Pascalban a GetDate, amely a GetTime-hoz hasonlóan a Dos unit része.

```
Program ElsoProgram_D;
  Uses
    Dos,Crt;
  Var
    O,P,M,m100:Word;
    EO,EP:Word;
    EM:LongInt;
    E,H,N,HN:Word;                    ← E:év, H:hó, N:nap, HN:a hét napja
  Const
    cim:String='A ma eltelt idő';
Begin
  ClrScr;
  Writeln(cim:40+(Length(cim) Div 2));
  Writeln;
  GetTime(O,P,M,m100);
  EO:=O; EP:=60*EO+P; EM:=60*EP+M;
  Writeln('Óra:',EO,' perc:',EP,
          ' másodperc:',EM);
  Writeln;                            ← Egy üres sor kiírása.
  GetDate(E,H,N,HN);                  ← A mai dátum bekérése. (Dos-beli)
  Write('A mai nap:',E,'.',H,'.',N);  ← A dátum változatlan kiírása, sor emelés nélkül.
  If HN=0 then Writeln(' vasárnap');
  If HN=1 then Writeln(' hétfő');
  If HN=2 then Writeln(' kedd');
  If HN=3 then Writeln(' szerda');    ← A HN függvényében a megfelelő nap kiírása, folytatólagosan.
  If HN=4 then Writeln(' csütörtök');
  If HN=5 then Writeln(' péntek');
  If HN=6 then Writeln(' szombat');
  ReadKey;
End.
```

Végezzük el ismét a program nyomon követését! Azt tapasztaljuk, hogy lehet bár hétfő, s így a második feltétel teljesül, vagyis a hétfő kiírásra kerül, a számítógép rendületlenül vizsgálgatja a következő 5 feltételt, minden „haszon” nélkül. [5. ábra]




**5. ábra:** Az ElsoProgram_D nyomkövetése.
Látható, hogy a vezérlés éppen a 33. sornál tart (az fog következni), és a felhasználói képernyő (User Window), amelyen már olvasható a „hétfő” szöveg.

# 5. Ötödik lépés

## 5.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől? Adjuk meg a napi dátumot, és mondjuk meg, milyen nap van ma! Ezeket is igényesen!

Programozási újdonság: hatékonysági megfontolás. [4. ábra]

## 5.1. Specifikáció

Ugyanaz, mint előbb.

## 5.2. Algoritmus

A hatékonyság növelését az algoritmikus nyelv többirányú elágazása biztosítja.

```
Program ElsőProgram_E:
  Változó
    Ó,P,M:Egész
    ÉÓ,ÉP,ÉM:Egész
    É,H,N,HN:Egész
  Konstans
    cim:Szöveg='A ma eltelt idő';
  Idő(Ó,P,M)
  ÉÓ:=Ó; ÉP:=60*ÉÓ+P; ÉM:=60*ÉP+M
  Ki:ÉÓ,ÉP,ÉM
  Dátum(É,H,N,HN)
  Ki:É,H,N [nincs soremelés]
  Elágazás
    HN=0 esetén  Ki:' vasárnap'
    HN=1 esetén  Ki:' hétfő'
    HN=2 esetén  Ki:' kedd'
    HN=3 esetén  Ki:' szerda'
    HN=4 esetén  Ki:' csütörtök'
    HN=5 esetén  Ki:' péntek'
   egyéb esetben Ki:' szombat'
  Elágazás vége
Program vége.
```

← *A* `HN` *függvényében a megfelelő nap kiírása, többirányú elágazással.*

A többirányú elágazás szemantikája: vizsgáld meg az 1. feltételt; ha az igaz, akkor hajtsd végre a hozzátartozó **`esetén`** kulcs-szó mögötti utasításokat, majd lépjél ki az elágazásból, azaz az **`Elágazás vége`** utáni utasítás következik; ha a feltétel nem igaz, akkor vizsgáld meg a következő feltételt és tégy úgy, amint azt előbb leírtuk.

## 5.3. Kód

A kódolás az `If` utasítás azon alakjára épít, amelynek van a feltétel nem teljesüléséhez is egy ága, nincs meg ugyanis a Pascalban az algoritmusban használt többirányú elágazás. Ezt a teljesebb `If` utasítást ágyazzuk egymásba 6-szor.

```
Program ElsoProgram_E;
  Uses
    Dos,Crt;
  Var
    O,P,M,m100:Word;
    EO,EP:Word;
    EM:LongInt;
    E,H,N,HN:Word;
  Const
    cim:String='A ma eltelt idő';
Begin
  ClrScr;
  Writeln(cim:40+(Length(cim) Div 2));
```

```
  Writeln;
  GetTime(O,P,M,m100);
  EO:=O; EP:=60*EO+P; EM:=60*EP+M;
  Writeln('Óra:',EO,' perc:',EP,
          ' másodperc:',EM);
  Writeln;
  GetDate(E,H,N,HN);
  Write('A mai nap:',E,'.',H,'.',N);
  If HN=0 then Writeln(' vasárnap')+
  else if HN=1 then Writeln(' hétfő')+
  else if HN=2 then Writeln(' kedd')+
  else if HN=3 then Writeln(' szerda')+
  else if HN=4 then Writeln(' csütörtök')+
  else if HN=5 then Writeln(' péntek')+
  else Writeln(' szombat');
  ReadKey;
End.
```

← *A* `HN` *függvényében a megfelelő nap kiírása.* `If`*-ek egymásba ágyazása.*

# 6. Hatodik lépés

## 6.1. A feladat

Hány (befejezett) óra, perc és másodperc telt el éjféltől? Adjuk meg a napi dátumot, és mondjuk meg, milyen nap van ma! Ezeket is igényesen!

Programozási újdonság a kódolásban van: ennél az algoritmusnál felhasználható a Pascal nyelv egy kevésbé általános elágazás fajtája, a `case` utasítás.

## 6.1. Specifikáció

Ugyanaz, mint előbb.

## 6.2. Algoritmus

Ugyanaz, mint előbb.

## 6.3. Kód

Figyeljük meg a `case` többirányú elágazás specialitását!

```
Program ElsoProgram_F;
  Uses
    Dos,Crt;
  Var
    EO,EP:Word;
    EM:LongInt;
    E,H,N,HN:Word;
  Const
    cim:String='A ma eltelt idő';
Begin
  ClrScr;
  Writeln(cim:40+(Length(cim) Div 2));
  Writeln;
  GetTime(O,P,M,m100);
  EO:=O; EP:=60*EO+P; EM:=60*EP+M;
  Writeln('Óra:',EO,' perc:',EP,
          ' másodperc:',EM);
  Writeln;
  GetDate(E,H,N,HN);
  Write('A mai nap:',E,'.',H,'.',N);
```

```
  Case HN of
    0:  Writeln(' vasárnap');
    1:  Writeln(' hétfő');
    2:  Writeln(' kedd');
    3:  Writeln(' szerda');
    4:  Writeln(' csütörtök');
    5:  Writeln(' péntek');
   else Writeln(' szombat');
  End;
  ReadKey;
End.
```

← *A* HN *függvényében a megfelelő nap kiírása,* case*-zel.*