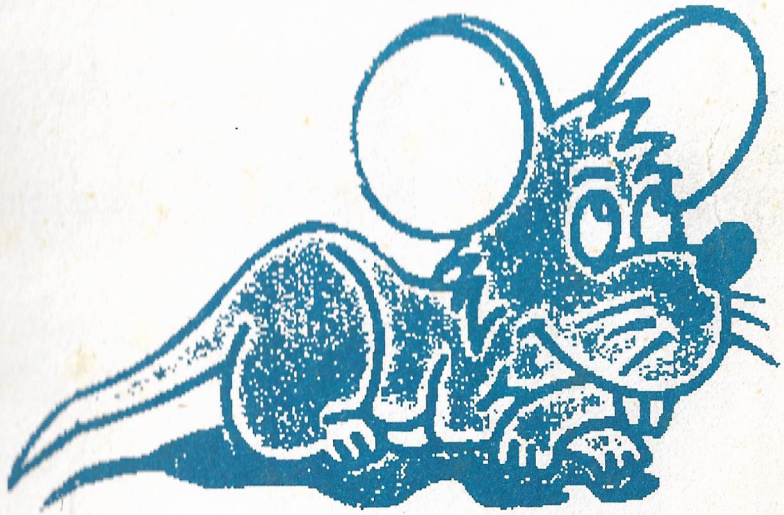
ABC STÚDIÓ BT.

Schneider Ferenc

A WORD PROGRAMNYELV



Békés Megye Képviselőtestülete
Pedagógiai Intézete

1991

Írta: *Schneider Ferenc*

Szerkesztő: *Povázsay Mihály*

A jegyzet a KFA pályázat támogatásával készült

A jegyzet megjelenését a Budapest Bank

Készült a Békés Megye Képviselőtestülete Pedagógiai Intézete sokszorosítójában 1000 példányban

Felelős kiadó: dr Kincses László igazgató

Forgalmazza és terjeszti az **APC-Stúdió BT.**
5700 Gyula, Kerék u. 9.

Schneider Ferenc

A WORD PROGRAMNYELV

Bevezető

A jegyzet elsősorban azoknak készült, akik szeretnének a számítógépek programozásának rejtelmeibe belekóstolni. Kezdőknek szól, olyanoknak, akik soha nem írtak egyetlen programot sem, de kíváncsiak arra, miként készülnek a programok!

Joggal vetődik fel a kérdés, mi is az a WORD programnyelv?

A programnyelv azért született, hogy fejlessze a problémamegoldó készséget, elsősorban gyermekeknél! A WORD egy magyar nyelvű, mondatszerkezetű programozási nyelv, mely sokkal közelebb áll a gyermekekhez és a felnőttekhez is, mint az angol nyelv. Az elképzelés már régen megszületett, hogy így lenne célszerű az elemi programszerkesztést tanítani. Az ötlet tehát nem az én érdemem!

A jegyzetben található feladatok elsősorban gyerekeknek szólnak, 9-12 éveseknek. Ez azonban nem zárja ki, hogy az érdeklődő felnőttek ne ismerkedhetnének meg a programozás elemi szabályaival. A cél gyakorlati megvalósításának segédeszköze a WORD, azaz SZÓ programnyelv.

A programnyelv IBM-PC, C+4, bővített C-16, C64, C128, és TVC gépeken egyaránt rendelkezésre áll. Bármely gépbe is tölti be a programot, a parancsok és az utasítások megegyeznek. Ha szüksége van a programnyelvre, akkor a jegyzetből kivágja a megrendelőlapot, és az adathordozó áráért (mágneslemez) elküldjük a programot. A füzettel együtt a program használati jogát is megvásárolta!

A program eredetileg a szakosított tantervű számítástechnika oktatásához lett kifejlesztve 9 éves gyermekeknek. Sikerrel használhatja azonban mindenki, aki a programozás elemi lépéseivel, a frontális programozási módszerrel szeretne megismerkedni!

Változók

Tapasztalhattuk, hogy a programokkal a számítógép mennyi mindenre felhasználható.

Itt az ideje, hogy mi is megpróbálkozzunk programok készítésével. Előtte azonban meg kell értenünk néhány igen fontos fogalmat!

Kezdjük talán egy példával.

A boltban mennyibe kerül:

Egy liter tej = _____
Egy kiló kenyér = _____
A kedvenc fogkrémed = _____
A kedvenc csokid = _____

Ha a mértékegységeket elhagyjuk és csak a lényegre törekszünk, így is leírhatnánk:

tej = _____
kenyér = _____
fogkrém = _____
csoki = _____

Egyfajta hozzárendelést végeztünk! A tejhez, a kenyérhez, ... egy-egy értéket rendeltünk, méghozzá számokat. Nézzünk még egy példát:

Pétert elküldte anyukája a boltba, és megkérte, hozzon 2 kiló krumplit. A boltban a krumpli kilója 15 Ft volt. Mennyit fizetett Péter a pénztárnál? Vegyük azokat az adatokat, amiket ismerünk:

Krumpli = _____
Ára = _____
Fizet = _____

Nézzük, hogyan számíthatjuk ki?

```
Fizet = Krumpli * Ára
```

A Fizet-hez hozzárendeljük a Krumpli szorozva az Árá-val kifejezést.

Hozzárendelés

Ha játékosan akarnánk a feladatot megoldani, szükségünk van három fiókra. A játékban kössük ki, minden fiókban csak egy szám lehet. Az egyes fiókokat meg kell különböztetnünk egymástól, különben összekeverjük őket. Ragasszuk rá az egyikre "krumpli"; a másikra "ára"; a harmadikra "fizet". Az első kettőbe helyezzünk be egy-egy cédulát, amelyekre felírjuk az adatokat, azaz a számokat. A harmadik fiók cédulájára pedig az általunk kiszámított érték kerül. Mennyi lesz ez?

_____

Nézzük, hogyan kapcsolódik mindez számítógépünkhöz!

Tolmácsprogramunk arra is alkalmas, hogy tetszőleges számú fiókot helyezzünk el. (Úgy képzeld el, mint egy sokfiókos szekrényt.) Itt is minden fiókban csak egy szám lehet. A fiókokat nevük alapján különböztetjük meg egymástól. Ezeket a fiókokat **VÁLTOZÓKNAK**-nak nevezzük.

Változók

A változók tartalma tetszőlegesen módosítható, cserélhető. Ha egy szám tipusú változónak korábban nem adtunk értéket, annak tartalma nulla.

Azért fontos hangsúlyozni azt, hogy szám tipusú, mert más (tehát nemcsak szám) változók is vannak! Olyan fiókot is használhatnánk, amelybe szöveget helyezünk el. Ezért a fiókjainkat különböző tartalmuk miatt (szám vagy szöveg) határozottan meg kell tudnunk különböztetni.

Attól függően, hogy milyen tipusú adatot (számot vagy szöveget) tárolunk fiókjainkban, beszélünk **szám tipusú változók**ról, vagy **szöveges tipusú változók**ról.

Adatok

A számok fontos szerepet töltenek be életünkben. Számokkal fejezzük ki, hány évesek vagyunk, mennyibe kerül a kedvenc gyümölcsünk. Ezek mind-mind adatok!

* **Csoportosítsuk az adatokat tipusuk szerint! Húzd alá a megfelelő választ!**

Írd le, hogy melyik évben születtél? _____

Szám **Betű** **Betű és szám**

Írd le, hogy melyik városban születtél? _____

Szám **Betű** **Betű és szám**

Írd le, a pontos lakáscímedet!

_____

Szám **Betű** **Betű és szám**

A számokat, vagy a szám tipusú adatokat numerikus értékeknek, vagy numerikus adatok-nak is emlegetik. A szám tipusú változókat pedig numerikus változók néven sem árt, ha ismered.

* **Sorolj fel numerikus adatokat !**

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

A W-nyelv mint tolmácsprogram

A mai órától egy új tolmáccsal (programnyelvvel) fogunk megismerkedni. Segítségével magyar nyelven készíthetünk programokat magunk is. ùj tolmácsunk szemmel láthatóan ért magyarul, pl.: a képernyőre kiírt üzenetek is magyar nyelvűek.

Ennek a tolmácsprogramnak a használatakor nem szabad elfelejtened a következő szabályt:

Minden parancs nagybetűvel kezdődik! (Ezt a SHIFT és a szükséges billentyű egyidejű lenyomásával érheted el.)

Nézzük meg először, hogyan számolhatunk új tolmácsunkkal.

```
Szamold ki
```

Sajnos gépünk eléggé hadilábon áll az ékezetes betűkkel, körülményesen, vagy egyáltalán nem ismeri. A parancs segítségével szorozhatunk(*), oszthatunk(/), összeadhatunk(+), kivonhatunk(-). Nézzünk egy példát!

```
Szamold ki 1O+5
```

Ha jól dolgoztunk, akkor a képernyőn megjelenik az eredmény. Végezd el az új tolmácsprogrammal a következő műveleteket!

4+3 = _____ 12-5 = _____

5*6 = _____ 2O/5 = _____

* Írd le, milyen műveleteket végeztél el a tolmácsodda!

Parancs és utasítás közötti különbség

Nézzük meg mi a különbség a parancs és az utasítás között! (Az eddigiekben mi csak a parancsokat használtuk.)

Parancs

- **A parancsokat gépünk azonnal végrehajtja**, ha a parancs végrehajtható!
- **A programok** azonban **utasításokból állnak**.
- A **gépünkbe betöltött programot csak akkor tudtuk elindítani**, használni, **ha erre külön parancsot adunk.** (Ez a BASIC-ben a RUN parancs volt.)

Nézzük most az utasításokat.

Utasítás

- **Az utasításokat gépünk megjegyzi**! Csak akkor hajtja végre a tolmács, ha erre külön parancsot adunk.
- Az utasításokat nem gépelhetjük be össze-vissza, hanem meghatározott sorrendben.

Nézzünk egy példát!

Ha az iskolából hazaindulsz, mielőtt felveszed a "kinti", utcai cipődet, le kell venned az iskolai, "benti" cipődet. Ruhadarabjaidat meghatározott sorrendben veszed fel. Matematikailag nem kifejezhető a sorrend és az sem, milyen színű, fazonú ruhát veszel fel.

Ezt a matematikailag nem kifejezhető sorrendet, **logikai sorrend**nek nevezzük! A logikai sorrendet elemi lépésekből építjük fel. Pl.: A cipő befűzése. (Gondold végig, milyen elemi lépésekből áll össze, amíg a cipő a lábadra kerül.)

A program is elemi lépésekből áll! **Az elemi lépéseket utasításoknak nevezzük.**

Elemi utasítások logikus sorozata alkotja a programot!

A program utasításai mindig sorszámmal kezdődnek! A sorok sorszáma a végrehajtás sorrendjét határozza meg.

Nézzünk meg egy egyszerű példát:

* Írjuk le vázlatosan egy hétköznapi reggelünket.

1 Felkelek.
2 Mosakodok.
3 Öltözködök.
4 Reggelizek.
5 Elindulok az iskolába.
6 Megérkezek az iskolába.
7 Vége.

Sorszámozásunk azonban így nem célszerű! Ha ugyanis kifelejtettünk valamit, az egészet kezdhetnénk elölről. Célszerűbb tehát 10 -től tizesével növekvő sorrendben sorszámoznunk.

10 Felkelek.
20 Mosakodok.
30 Öltözködök.
40 Reggelizek.
50 Elindulok az iskolába.
60 Megérkezek az iskolába.
70 Vége.

Így lehetőségünk van a megfelelő sorszám beírásával módosítani a vázlatot, ha szükséges pl.:
35 Elpakolok.

* Vázlatunkból hiányzik a búcsú családunktól. Írd le megfelelő sorszámmal ellátva!

Írd le sorszámmal ellátva az esti programodat!

A W-nyelv alaputasításai 1.

Nézzük, új tolmácsunk milyen utasításokkal rendelkezik, mi a feladata ezeknek.

Értékadás

Valtozok: xx (,yy).

Az utasítás segítségével "xx" (és "yy") nevű változókat (fiókokat) jelölünk ki programunkban.

Minden program ezzel a szóval kell, hogy kezdődjön.

A tolmácsnak szüksége van arra, hogy megmondjuk milyen nevű fiókokat fogunk programunkban használni. Ezért is fontos, hogy **a programot mindíg előre megtervezzük!**

Legyen nev = szam.

Ez az értékadó utasítás. Az utasítással adhatunk értéket egy-egy változónak (fióknak). Használata kötelező minden értékadáskor.

A "szam" egy numerikus adat.

Pl.: Legyen nev=5.

Kiíratás

Ird ki nev.

Az utasítás a képernyőre írja a "nev" nevű változó értékét.

Ha az előző értékadást használjuk, a képernyőre kiíródik az 5, mert ennyi a "nev" nevű változónk tartalma.

Műveletvégzés

Tolmácsunk segítségével műveleteket is végezhetünk a programban.

Add	= összeadás
Vond	= kivonás
Szorozd	= szorzás
Oszd	= osztás
Vege	= vége a programnak

Ezek után nézzük meg, mit kezdhetünk az eddig tanultakkal. Nézzünk meg egy egyszerű feladatot! A feladatnál és annak megoldásánál ne azt nézd, hogy ez mennyire egyszerű, hanem azt, hogyan oldjuk meg tolmácsprogramunk segítségével!

* Anna vett egy ceruzát 3 Ft-ért és egy radírt 4 Ft-ért. Mennyit fizetett a pénztárnál?

Milyen adatokat ismerünk:

ceruza	=	3
radír	=	4
fizet	=	?

Hogyan számoljuk ki:_____

Most nézzük meg, hogy lesz ebből program!

Megoldásminta 1

1O Valtozok: ceruza, radir, fizet.
2O Legyen a ceruza = 3.
3O Legyen a radir = 4.
4O Add a ceruzat a radirhoz ,tedd fizetbe.
5O Ird ki fizet.
6O Vege.

Kész is a program. Nézzük meg, melyik sor mire való!

- A tízes sorban gondoskodunk a szükséges változók (fiókok) elnevezéséről. (Szó volt róla korábban, ha nem emlékszel rá, olvasd el a Változók című fejezetet.)
- A húszas sorban értéket adunk a "ceruza" nevű változónak.
- A harmincas sorban a "radir" nevű változó kap értéket.
- A negyvenes sorban elvégezzük az összeadást, az eredményt a "fizet" változóba tesszük.
- Az ötvenes sorban kiíratjuk az eredményt.
- A hatvanas sor jelzi; itt a program vége.

Akár be is gépelhetjük programunkat!

Gépünk csak annyit fog tudni, amennyire a tolmácsprogramon keresztül megtanítjuk! Ha valamit kifelejtesz, vagy nem tudsz, programod is hibásan fog működni!

A program "indítása" előtt azonban meg kell ismerkednünk néhány újabb paranccsal!

A programot begépelve még nem használhatjuk! Előtte a számítógép számára is le kell fordítanunk a programot. Erre szolgál a következő parancs.

Fordítás

Fordits

Tolmácsunk most lefordítja a programot! Ez is parancs, hisz tolmácsunk azonnal végrehajtja! A fordításra azért van szükség, mert a számítógép nem tudna mit kezdeni az általunk begépelt programmal. A tolmácsnak kell a gép számára is érthetővé tenni! Erre utasítjuk a tolmácsot a parancs segítségével!

Program indítása, futtatása

A lefordított hibátlan program vérehajtására szintén parancsot kell adnunk tolmácsunknak. Különben hiábavaló volt a munka, nem tudjuk megnézni, mit is csinál programunk!

Fuss

A parancs elindítja a lefordított programunkat! Ez azt jelenti, hogy lépésről lépésre végrehajtja azokat az utasításokat, amik programunkat alkotják.

Program listázása

Ha a begépelt programot szeretnénk megnézni, szükségünk van egy olyan parancsra is, amely kiírja a képernyőre a programot. Erre főleg akkor van szükség, ha valamit hibásan gépeltünk be, és a fordításkor a tolmács hibát jelzett. Ha ezt a parancsot használjuk, láthatjuk az egész program listáját.

Lista

A parancs kiírja a képernyőre a W-nyelvű programot.

Program törlése

A következő parancsot akkor kell használnunk, ha gépünkbe már van egy program, de egy újabbat szeretnénk begépelni.

Ujra

A parancs hatására tolmácsunk törli a gépben levő programot.

Problémaelemzés

Egy probléma megoldásához a következő sorrendet kell betartanunk:

1. Elemezzük (értsük meg) a feladatot.
2. Határozzuk meg, hogyan oldhatjuk meg.
3. Írjuk meg a programot a füzetbe.
4. Ellenőrizzük, működik-e a program.
5. Ha hibás, menj vissza az 1-es pontra, különben tovább.
6. Gépeljük be a számítógépünkbe.
7. Fordítsuk le a tolmáccsal.
8. Indítsuk el.
9. Próbáljuk ki olyan adatokkal, amelyekkel egyszerűen ellenőrizhetjük a kapott eredmény helyességét!

A Word nyelven írt programsorok 70-78 karakternél hosszabbak nem lehetnek! Azért nem lehet pontosan megmondani, mert ez géptipusonként eltérhet. Számold meg és írd le hány jelet (karaktert) írhatsz egy W-nyelvű sorba! _____

A mai órától hoznod kell a matematika könyvedet az órára!

FELADATOK:

* Írd össze milyen **parancsait** ismered a W-nyelvnek, és ezek a parancsok mit jelentenek!

* Írd össze milyen **utasításait** ismered a W-nyelvnek és ezeket mikor alkalmazzuk!

* Pista elment a boltba és vett 4 kilógramm almát. Ára 15 Ft/kg. Mennyit fizetett a pénztárnál?
Készíts W -nyelvű programot a feladatból!

Feladatterv:

Az első saját programom

Írta:_____

Dátum : _____

A feladatok értelmezése 1.

Ez a fejezet segítséget ad a feladatok pontos értelmezéséhez. Olyan "általános recept" nincs, amivel minden feladat egyformán megoldható. Vannak azonban olyan általános szabályok, amelyek betartásával a feladatok egyszerűbben oldhatók meg. Az előző órán már megbeszéltünk egy általános feladatmegoldást. Most nézzük meg részletesebben.

Az első és legfontosabb a feladat pontos elolvasása.

Ezt követi a feladat pontos értelmezése. Meg kell értened mi a feladat! Amíg nem érted, nem tudsz azon sem gondolkozni, hogyan lehetne megoldani.

Gyakran hallani kétségbe esve diákoktól: - "én ezt nem értem, nem tudom megoldani"! Pedig csak arról van szó, hogy nem olvasta el figyelmesen, azaz nem értette meg a kisdiák mi a feladat. Ahhoz lehetne hasonlítani, ha otthon megkérnek, vásárolj 2 liter tejet, és te 2 pár zoknival lepnéd meg a családot tej helyett. Összegezve: rengeteg bonyodalom származik abból, ha nem értettük meg, mit is kell csinálnunk!

Mit kell ilyenkor tenni?

- Olvassuk el ismét figyelmesen és lassan a feladatot!

- Olvassuk el úgy a feladatot, hogy meghatározzuk milyen adatokat ismerünk, és sorra vesszük ezeket az értékeket. Célszerű, ha az egyes adatokat kis téglalapokkal jelöljük, a feladatnak megfelelően egyszerűen elnevezzük, majd a téglalapba beírjuk ezek tartalmát is.

- Határozzuk meg a keresendő értéket! Nevezzük el ezt is!

Ha valaki betartotta az iménti szabályokat, akkor eljutott odáig, hogy meghatározta, **mit ismer és mit keres**. A neheze most következik.

- Meg kell határozni, hogyan lehet az ismert adatokból a keresett adatot, vagy adatokat meghatározni.

Ilyenkor kezdődnek a bonyodalmak!

El kell döntenünk; **elég ismert adat áll-e rendelkezésünkre, hogy a feladatot megoldhassuk**. A feladatnak ugyanis **két nagy csoportja van!**

1. Az ismert adatokból a keresett adat (eredmény), vagy adatok (eredmények) azonnal meghatározhatók. Az ilyen tipusú feladatokat **közvetlenül megoldható feladatok**nak nevezzük.
2. Az ismert adatokból a keresett adat (eredmény), vagy adatok (eredmények) nem határozhatók meg azonnal. Ilyenkor valamilyen közbülső műveletre van szükségünk. Az ismert adatok felhasználásával egy olyan adatot (részeredményt) kell előállítanunk, amelynek felhasználásával a feladat már megoldható. Az ilyen tipusú feladatokat **közvetetten megoldható feladatok**nak nevezzük.

Most kanyarodjunk vissza a .feladat megoldásához!

- Vizsgáljuk meg, elegendő adatot ismerünk-e a feladat megoldásához!

- Amennyiben szükséges, próbáljuk meghatározni milyen részeredményre van szükségünk a feladat megoldásához.

Vegyük az egyszerűbb esetet. Mondjuk azt, hogy egy olyan feladattal van dolgunk, ahol az ismert adatokból azonnal meghatározható a keresett érték. De hogyan?

Vannak ugyanis olyan diákok is, akik egy szöveges feladat megoldásánál az összeadás , kivonás, szorzás, osztás műveletét úgy használják, mintha totóznának. Az esetek többségében persze nem találják el a helyes műveletet. A nagyobbik baj az, hogy a **hibás eredményeket** sikeres megoldásként könyvelik el.

Milyen segítséget, tanácsot lehet adni ilyen esetben?

-Legegyszerűbb, ha végiggondoljuk logikusan, lehetséges-e a kapott eredmény.

-Másik megoldás az ellenőrzés. Ez sokkal megbíztatóbb, eredményesebb.

Nézzük meg egy feladaton keresztül, a gyakorlatban mit is ér ez a tudomány!

* Csillának 20 Ft-ja volt. Bement a boltba és 10 Ft-ért vett csokit. Mennyit kapott vissza a pénztárnál?

- Olvassuk el figyelmesen a feladatot még egyszer!
- Vegyük sorra mit ismerünk. Nevezzük el az egyes adatainkat, és jelöljük kis téglalapokkal!

Mit ismerünk?

forint	20
csoki	10

Mit keresünk?

vissza	?

Hogyan határozhatjuk meg az ismert adatokból a keresett adatot?

Itt azonban megállunk egy **tanulságos próbálkozásra**. Tételezzük fel, hogy fogalmunk sincs, hogyan kell megoldani a feladatot!

Kezdjünk el tehát próbálkozni.

- *Első kísérletként próbáljuk ki, hogy a két ismert adatot összeadjuk.*

Azaz: vissza = forint + csoki
Helyettesítsünk be: vissza = 20 + 10
Eredmény: vissza = 30

Most gondoljuk végig a feladatot! Lehetséges-e, hogy a 20 Ft-ból a csoki vásárlása után 30 Ft-ot kapunk vissza a pénztárnál? Ez a megoldás első ránézésre is **hibás!!!** Ilyen durva hibánál nem kell bizonyítanod miért rossz!

- *Második próbálkozásunk legyen az, hogy a két számot összeszorozzuk.*

Azaz: vissza = forint * csoki
Helyettesítsük be: vissza = 20 * 10
Eredmény: vissza = 200

Gondoljuk végig a feladatot és nézzük meg a kapott eredményt! Lehetséges-e hogy 20 Ft-ból, ha a pénztárnál 10 Ft-ot fizetünk, 200 Ft-ot kapunk vissza? Ez a megoldás is **hibás!** Ha a hibákat lehetne fokozni ez a megoldás hibásabb az előzőnél is.

- *Harmadjára próbálkozzunk az osztással, talán ez segít rajtunk.*

Azaz: vissza = forint / csoki
Helyettesítsük be: vissza = 20 / 10
Eredmény: vissza = 2

Gondoljuk végig a feladatot és nézzük meg a kapott eredményt. Lehetséges-e, hogy 20 Ft-ból, ha a pénztárnál 10 Ft-ot fizetünk, 2 Ft-ot kapunk vissza? Ez már ravaszabb kérdés. Ha valaki nem számolja ki az eredményt, akkor csak az **ellenőrzés** segíthet a kérdés tisztázásában.

Ellenőrizzük a megoldást!

Nézzük meg tehát, hogyan ellenőrizhetjük az eredményt!

Ellenőrzéskor mindig az **eredeti szöveges feladatba helyezzük vissza a kapott eredményt.** A feladatunk az általunk kapott megoldással így néz ki:

Csillának 20 Ft-ja volt. Bement a boltba és 10 Ft-ért vett csokit. A pénztárnál 2 Ft-ot kapott vissza.

Aki figyelmesen megvizsgálja az adatokat, az azonnal láthatja, valami nincs rendben. Csillát alaposan becsapták! A visszajáró pénznek és a csoki árának (ha ezt a kettőt összeadjuk) eredményként az eredeti összeget kell kapnunk. Ennél a megoldásnál azonban láthatjuk, hogy ez sehogyan sem akar egyezni.

Azaz: forint = csoki + vissza
Behelyettesítve: 20 = 10 + 2
Igaz ez az állítás?

A megoldás tehát hibás, akár csak a korábbiakban!

- *Negyedik kísérletként próbálkozzunk a kivonással.*

Azaz: vissza = forint - csoki
Helyettesítsük be: vissza = 20 - 10
Eredmény: vissza = 10

Gondoljuk végig a feladatot és nézzük meg a kapott eredményt. Lehetséges-e, hogy 20 Ft-ból, ha a pénztárnál 10 Ft-ot fizetünk, 10 Ft-ot kapunk vissza? Válaszként hívjuk segítségül az ellenőrzést!

Csillának 20 Ft-ja volt. Bement a boltba és 10-Ft-ért vett csokit. A pénztárnál 10 Ft-ot kapott vissza.

Azaz: forint = csoki + vissza
Behelyettesítve: 20 = 10 + 10
Igaz ez az állítás?

Sokan túl egyszerűnek találják a feladatot és feleslegesnek a három eredménytelen kísérletet. Vannak azonban olyan "egyszerűnek látszó" feladatok is, amelyek néha alaposan próbára teszik még a legjobb tanulókat is. Tanácsként, segítségként oldottuk meg a feladatokat így, mindenki okulására.

Kitűnik minden megoldásból a lényeg: **GONDOLKOZZ!**

A sikeres, hibátlan programkészítés nem egyszerű feladat! Sok-sok problémát kell megoldani, amíg valaki terv nélkül is képes azonnal felismerni a megoldáshoz vezető helyes utat.

Minden esetben készíts megoldási tervet!
Az elkészült terv megoldási menetét ellenőrízd le!
Ha hibát fedeztél fel, javítsd ki és kezd elölről az ellenőrzést!

A probléma megoldásához vezető út meghatározását algoritmusnak nevezzük.

Az algoritmus szerkesztése nem egyszerű mesterség!

A W-nyelv utasításai 2.

Programjaink elkészítésénél sokszor tapasztaltuk, hogy a feladatok tipusa bizony gyakran ismétlődik. Gondolj csak a következő feladatokra:

* Anna 3 Ft-ért ceruzát, 4 Ft-ért radírt vásárolt. Mennyit fizetett?
* Péter 5 Ft-ért ceruzát, 12 Ft-ért radírt vásárolt. Mennyit fizetett?

A feladatok megoldási menete azonos!

Ha lenne olyan utasításunk, amellyel az egyes változók értékét a billentyűzetről kérhetnénk be, olyan programokat írhatnánk, amelyekkel az előző feladatokhoz elegendő lenne egyetlen programot készíteni. Ezeknek a feladatoknak a megoldása azonos, csak az adatok változnak.

Adatbevitel a billentyűzetről

Kerd be "XX".

Az utasítás hatására a gép kiírja a képernyőre a változó nevét (XX) és egy kérdőjelet (?). Gépeljük be azt a számot, amit a változó értékének szánunk, és nyomjuk meg a "RETURN" billentyűt. Nézzünk egy példát az alkalmazásra:

Készítsünk programot, amely összead két számot!

Mit ismerünk?

Első szám	
Második szám	

Mit keresünk?

Eredmény	?

A megoldás algoritmusa:

Eredmény = Első szám + Második szám

Nézzük a programot! (A változókon rövidítünk.)

```
10 Valtozok: elso, masod, eredmeny.
20 Kerd be az elsot.
30 Kerd be a masod.
40 Add elso masodhoz ,tedd eredmenybe.
50 Ird ki az eredmenyt.
60 Vege.
```

Próbáld ki a programot!

* Az első két feladat megoldásához készíts feladattervet és programot az új utasítás felhasználásával!

Feladatterv:

* Készíts programot, amely kiszámolja a menzadíj alapján, a havi étkezési díjat.

Feladatterv:

Feladatok értelmezése 2.

Rejtett adatok

Az eddigi "egyszerűen" megoldható feladatok mindinkább nehezebbek lesznek. Vannak olyan adatok, amelyeket nem ismerünk. Mihez kezdjünk ilyenkor, ki segít a szegény diákon? Nézzük csak az előző szabályt!

- Olvassuk el ismét figyelmesen a feladatot!

- Majd ismét, de most már vegyük sorra, hogy milyen adatokat ismerünk és állapítsuk meg ezek értékét. Célszerű, ha az egyes adatokat a feladatnak megfelelően egyszerűen elnevezzük, majd egyenlőség jelet téve, mögé írjuk ezek tartalmát is.

Ezt követően;

- Nevezzük el a keresendő értéket.

Majd a megoldás **algoritmusa** következik.

- Határozzuk meg, hogyan számolhatjuk ki a keresett értéket az ismert adatokból.

Nézzünk egy példát:

* Készítsünk programot, amely kiszámolja, egy hét hány óra.

Tartsuk be az előbbi szabályt! A figyelmes olvasó azonnal rádöbben, hogy itt egy olyan feladattal van dolgunk, amiben szemernyi ismert adat sincs. **Valóban nincs ismert adat?**

Egy hét hány nap? _____

Egy nap hány óra? _____

Mit ismerünk?

Az első adat:

nap [ ]

A második adat:

óra [ ]

Mit keresünk?

hét [?]

Ismert tehát a "nap" és az "óra"! Keressük azt, hogy egy hét hány óra? Meghatározható az ismert adatokból a keresett adat? A kérdésre csak gondolkodás után szabad válaszolni. Nem mindenkinek természetes a válasz. Próbáljunk meg segíteni!

hét = nap * óra

* A feladat megoldása innen már házi feladat!

HÁZI FELADAT:

Változók értékadása

Sokan felteszik a kérdést;

- **"Mikor használjuk a "LEGYEN" és mikor a "KERD BE" utasításokat."**

Igazából ilyenkor az a pontos válasz, hogy **az adott feladat határozza meg,** mikor kérjünk be, és mikor adjunk a programban értéket egy változónak. Ettől a választól azonban nem lett egy diák sem okosabb, pedik ez a válasz hibátlan! Nézzük meg közelebbről mit is jelent a válasz.

> Ha egy feladatban olyan adatokkal találkozunk, melyek értékét ismerjük, akkor egyértelműen látszik a válasz, a "LEGYEN" utasítást kell használni.

> Ha egy adat tartalmát nem tudjuk meghatározni, nem ismerjük annak lehetséges tartalmát sem, akkor a "KERD BE" utasítást kell használnunk.

A valóságban azonban nem ilyen egyszerű a helyzet. Ha az előző feladatot tekintjük, akkor célszerű, ha a feladat megoldásában csak az "óra" nevű adatot határozzuk meg a "LEGYEN" utasítással, mivel egy nap mindig 24 órából áll. A "nap" nevű adatot viszont célszerű, ha bekérjük a program használójától a "KERD BE" utasítással.

Mi az előnye ennek a megoldásnak? Igy nemcsak egy hetet, hanem tetszőleges napszámokat is át tudunk számolni órára (akár hónapot, vagy évet is).

Ez azonban nem azt jelenti, hogy nem használjuk a "LEGYEN" utasítást!

> A célszerűséget próbáljuk meg szem előtt tartani. A "LEGYEN" utasítást olyan adatoknál használjuk, amikor nincs értelme egy változó tartalmának a bekérésére. (Pl.: mértékegység, váltószám, stb. mivel ezek értéke állandó!)

A különböző matematikai problémák megoldása közben gyakran felmerül a kérdés; matematika-e a számítástechnika?

A tolmácsprogram segítségével megtanítjuk gépünket egy-egy feladat megoldására. Nem mi számolunk, hanem programunk segítségével gépünk számolja ki az eredményeket. A te kezedben számítógép van, amely pontosan kiszámolja az általad előírt műveletet. Nem fog hibázni, így az eredményben biztosak lehetünk!

Hogy mit írsz elő a gépnek, azt te határozod meg. Ha butaságra tanítod meg, butaságot hajt végre.

Képzelj el egy olyan robotot, amiben egy hibás program fut (a robotokat is számítógépek vezérlik) vagy egy korszerű telefonközpontot, ahol hibás a program. (Ha mondjuk felhívnád anyu munkahelyét, a pontos időt, vagy a telefondoktort kapcsolná.)

Programellenőrzés

Programjainkat tehát gondosan és pontosan kell elkészíteni. Az elkészült programot pedig feltétlen ellenőrizni kell. Az első ellenőrzéshez ritkán használjuk gépünket. Ennek az is oka, hogy a házi feladatok megoldásánál ritkán van lehetőség géphasználatra. Ilyenkor is ellenőrizni kell azonban a programot.

> A program ellenőrzését, a **program tesztelésének** nevezzük.

Ilyenkor sorról-sorra nekünk kell végrehajtani az utasításokat, mintha mi lennénk a számítógép. Csak azt és annyit szabad azonban végrehajtani, amit előírtunk! Ez a legegyszerűbb és legelterjettebb formája a program ellenőrzésének.

Az utasítások következetes végrehajtásával az esetleges kimaradt programsorok így azonnal pótolhatók, a fölösleges sorok elhagyhatók. **A végrehajtáshoz azonban adatokra van szükségünk.**

Adatok egyszerűsítése

Amikor egy probléma megoldásának helyességét ellenőrizzük (teszteljük) akkor ezt kétféleképpen is megtehetjük:

Egyik lehetőség, ha a feladatnak megfelelő adatokat veszünk alapul.

Ha az adatok túl nagy értékűek, akkor vállaljuk a kockázatot is, hogy esetleg elszámolunk valamit. Így előfordulhat, hogy a hibátlan programban keressük a hibát.

Másik lehetőség, ha a feladat engedi, adatként kis számokat használunk.

Természetesen nem egyszerűsíthetünk mértékegységeket, vagy váltószámokat! Célszerű, ha ilyenkor arányosan csökkentjük az értékeket.

Pl.:

```
20 Legyen a forint = 200.
30 Legyen az alma = 10.
.
.
.
```

Ellenőrzéshez azonban a következő értékeket használhatjuk:

```
20 Legyen a forint = 20.
30 Legyen az alma = 1.
.
.
.
```

Így egyszerűen ellenőrizhetjük (a hibás számolás lehetőségének csökkentésével), hogy programunk hibátlanul dolgozik-e. Nekünk ugyanis elsősorban a megoldásra, a feladat végrehajtására kell gépünket megtanítani, nem pedig számolni.

Most egy nagyon fontos részhez érkeztünk! Nekünk tehát az a dolgunk, hogy egy-egy feladat megoldásának az elemi lépéseire tanítsuk meg a gépet. Minden feladat számunkra egy-egy **probléma, melynek megoldási menetét nekünk kell megtalálni, meghatározni!** Olyan szabályt kell tehát keresnünk, amellyel a probléma (az egész feladat) megoldható. A szabály megfogalmazásánál az esetek többségében a kiinduló adatok számunkra érdektelenek (kivéve a mértékegységek, vagy a váltószámok).

Gépednek teljesen mindegy, hogy egy változó tartalma 23, vagy 2367. Az általunk megfogalmazott szabály szerint sorról-sorra végrehajtja a programot. Nekünk azonban nem mindegy milyen értékekkel számolunk.

A megfogalmazott szabályt, amellyel a probléma megoldható, algoritmusnak nevezzük! Erről már korábban is szó volt.

Az algoritmus véges számú, elemi utasítások logikus sorrendje.

Algoritmus

A számítógép nem képes (egyenlőre) algoritmusok (szabályok) önálló megfogalmazására. Meg kell tehát tanítanunk minden egyes probléma megoldási szabályára. A megoldás menetének ismeretét nevezzük algoritmusnak. Te magad is tapasztalhatod, hogy bizony nem egyszerű megtalálni azt a szabályt, amivel egy probléma hibátlanul megoldható.

* Soroljunk fel az eddig elkészült munkáinkból olyanokat, ahol az adat nem befolyásolja a megoldás algoritmusát!

_____

_____

* Soroljunk fel az eddig elkészült munkáinkból olyanokat, ahol az adat részben befolyásolja a megoldás algoritmusát!

_____

_____

A feladatok értelmezése 3.

Összetett algoritmusok

Az eddigi feladataink többsége közvetlenül megoldható feladat volt. A közvetetten megoldható feladatok algoritmusa bonyolultabb. A szabályt, mely a helyes megoldáshoz vezet, nehéz megtalálni. Itt különösen fontos az eddig tanultak betartása! Igen komoly segítséget jelenthet. Nézzük meg egy feladaton keresztül.

* Csillának 200 Ft-ja volt. Bement a boltba és 17 Ft-ért vett csokit, 38 Ft-ért fogkrémet. Mennyit kapott vissza a pénztárnál?

Olvassuk el figyelmesen a feladatot ismét!

Vegyük sorra mit ismerünk!

Nevezzük el az egyes adatainkat és jelöljük is téglalapokkal!

Mit ismerünk?

forint	200
csoki	17
fogkrém	38

Mit keresünk?

vissza	?

– Hogyan határozhatjuk meg az ismert adatokból a keresett adatot. Azaz milyen szabályt (algoritmust) kell készítenünk a feladat hibátlan megoldásához?

Ilyenkor általában mindenki azonnal elkezd számolni. Pedig, ha használjuk a fejünket, gyorsabban célhoz érünk. Már megbeszéltük, hogy a feladat megoldása szempontjából az adatok nem fontosak (hacsak nem mértékegység, vagy váltószám). Ennél a feladatnál egyik kizáró ok sem szerepel. Figyeljük meg, hogyan egyszerűsítettük a feladatot!

Mit ismerünk?

forint	20
csoki	1
fogkrém	3

Mit keresünk?

vissza	?

A megoldás szabálya (algoritmusa) ettől még semmit nem változott! Ugyanúgy kell megoldani, mint az előző változatot, de kevesebb hibánk lesz, ha számolnunk kell! Az adatokat is egyszerűbben kezelhetjük.

Ha az összes pénzünkből jelen esetben 20 Ft-ból költünk el valamennyit, akkor a vásárolt érték és a maradék pénzünk meg kell egyezzen, az eredeti összeggel. Ez kicsit bonyolultan hangzik, de igen egyszerűen belátható:

forint = csoki + fogkrém + vissza

A megoldás keresésénél láthatjuk, hogy a visszajáró összeget nem tudjuk azonnal (egyetlen művelettel) meghatározni.

- A megoldás egy lehetséges módja az, amit a pénztárnál is tapasztalunk. Összeadjuk a vásárolt árukat, jelen esetben a csokit és a fogkrémet. A kapott eredmény a fizetendő összeg. Ezt vonjuk le az összes pénzünkből, így kapjuk meg a visszajáró összeget. Ha ezt az utat választjuk, akkor az áruk összegének is kell egy változó a születendő programban.

forint	20
csoki	1
fogkrém	3
vissza	?

A megoldáshoz vezető út tehát a következő:

áru = csoki + fogkrém

vissza = forint - áru

Innen már csak a behelyettesítés van hátra:

áru = 1 + 3

vissza = 20 - 4

vissza = 16

Ellenőrizzük hibátlan-e az algoritmus!

forint = csoki + fogkrém + vissza

20 = 1 + 3 + 16

Végezzük el a műveletet és döntsük el igaz-e az állítás. Ezek után az eredeti adatokkal nyugodtan megírhatjuk a programot!

* A feladatnak van azonban egy másik megoldási módja is. Határozzuk meg közösen ennek az algoritmusát!

Feladatterv:

Nézzünk egy másik feladatot!

* Egy iskolába 200 tanuló jár. Ennek a fele és még 20 a lányok száma. Hány lány és hány fiú jár az iskolába?

Olvassuk el figyelmesen a feladatot még egyszer!

Vegyük sorra mit ismerünk! Nevezzük el az egyes adatainkat, és jelöljük kis téglalapokkal.

Mit ismerünk?

tanulók	200
fele	2
plusz	20

Mit keresünk?

lányok	?
fiúk	?

Hogyan határozhatjuk meg az ismert adatokból a keresett adatot. Azaz milyen szabályt (algoritmust) kell készítenünk a feladat hibátlan megoldásához?

A tanulók számát a lányok és a fiúk összege fogja adni!

tanulók = lányok + fiúk

Ebből csak a tanulók száma ismert. Hogyan határozhatjuk meg a lányok és a fiúk számát? Mielőtt erre rátérnénk nézzük meg, tudunk-e magunkon segíteni. Tudjuk-e egyszerűsíteni az adatokat, és hogyan?

Mit ismerünk?

tanulók	20
fele	2
plusz	2

Mit keresünk?

lányok	?
fiúk	?

Első lépésként a tanulók felét kell meghatároznunk!

fele = tanulók/2

A lányok számát úgy határozhatjuk meg, hogy a tanulók fele létszámához még pluszban hozzáadunk kettőt.

lányok = fele + plusz

A fiúk létszámának megállapításához két megoldás is lehetséges.

1. fiúk = tanulók - lányok

2. fiúk = fele - plusz

Tetszőlegesen választhatunk a két megoldás közül. Célszerűbb, ha a számodra érthetőbbet választod! Az algoritmus végleges meghatározása a megoldás. Az ellenőrzés a te feladatod!

Az algoritmusok szerkesztését nyugodtan nevezhetjük a gondolkodás iskolájának. Hétköznapi életünk is algoritmusok szövevényen keresztül zajlik, csak ez természetes, hozzászoktunk. Bizonyos esetekben egyszerűbbnek is látszik, mivel nem, vagy igen kevés matematikai műveletet tartalmaz. Mi is fogunk olyan algoritmusokat szerkeszteni, amelyekben egyáltalán nem, vagy csak kicsiny része lesz matematikai művelet.

Word programnyelv lehetőségei

Egyenlőre elsősorban **matematikai problémák**at oldunk meg, mert **a Word programnyelv ezt támogatja**, ilyen tipusu feladatok megoldására alkalmas.

Mindezek mellett még a matematika órán is kipróbálhatod az itt szerzett tapasztalataid.

Mintafeladat - megoldások

Ami kimaradt a W-nyelv utasításaiból 1 /kiegészítő/

Programjainkat használatuk során gyakran többször is le kellett futtatnunk, hogy kiszámoljuk az összes eredményt. Miért ne írhatnánk olyan programot, amely ezt magától is megteszi? Természetesen erre meg kell tanítanunk a programunkat!

Menj sorszam (-ra, -re)

Az utasítás hatására programunk végrehajtása az általunk megadott sorszámú sortól folytatódik. Nézzünk egy példát!

* Készítsünk egy programot, amely két szám különbségét írja ki a képernyőre!

Elemezzük a feladatot!

Első szám	
Második szám	
Eredmény	?

Eredmény = első szám - második szám

A WORD nyelvű program:

10 Valtozok: elso, masodik, eredmeny.
20 Kerd be az elsot.
30 Kerd be a masod
40 Vond ki elsobol masod ,tedd eredmenybe.
50 Ird ki az eredmenyt.
60 Menj 20-ra.
70 Vege.

Programunk így az eredmény kiírása után mindíg visszatér a 20-as sorra, azaz bekéri az első számot. Semmi értelme nem lenne visszatérnünk a 10-es sorra, hiszen a változókat (fiókjainkat) csak egyszer kell kijelölnünk. Ha a 30-as sorra mennénk vissza, akkor programunk csak a második számot kérdezné meg újra! Ez is hibás működést eredményezne, hiszen az első változó értékét csak egyszer kérdezné meg. Így mindig ebből az értékből vonná ki az újonnan megadott második értéket. Tanulságos lenne kipróbálni.

Hogyan lehetne megállítani programunkat? Így ugyanis "végtelen" sokszor ismétlődik az algoritmus. Nyomjuk meg a RUN/STOP, vagy IBM-PC gépen a "Ctrl + Break" billentyűt! Segítségével elérhetjük, hogy programunk megálljon. A továbbiakban számítógépünk "kész" parancsaink fogadására.

Az utasítást csak akkor alkalmazzuk, ha valóban szükséges többször is végrehajtanunk ugyanazt a programrészletet!

Nézzünk egy feladatot!

* Andrist elküldte anyukája a boltba dióért. Adott neki 200 Ft-ot.Az egyik boltban 1 kg-ot, a másikban 2 kg-ot, a harmadikban 3 kg-ot kapott volna ezért a pénzért. Mennyibe került a dió kilója az első, a második, és a harmadik boltban? Készítsük el a programot!

Feladatterv:

Ami kimaradt a W-nyelv utasításaiból 2 /kiegészítő/

A programok készítésénél gyakran lenne szükségünk olyan utasításra, amelynek felhasználásával dönthetünk az utasítások ismétléséről.

Döntés az algoritmusban és a programban

Nézzünk egy példát!

- Mikor dicsérnek meg otthon?
- Ha ötöst viszek haza, akkor megdicsérnek otthon!

Látható, hogy a válaszban **feltételhez kötött döntés** van!

Ha az állítás igaz, akkor ... .

< változó2

Ha a változó = változó2 ,akkor menj a sorszámra.

> változó2

Itt tehát két változót hasonlítunk össze, a <, >, = relációjelek segítségével. A változó1 és a változó2 azok a változónevek, amelyeket mi adunk meg. Tartalmukról is nekünk kell gondoskodnunk!

Alkalmazását nézzük meg egy feladaton keresztül!

* Anyu megkérte Csillát, írja össze mit kell vásárolni a boltban, és írja az áruk mellé az árakat is. Ha elkészült, adja össze és mondja meg, hány forintot kell magukkal vinni a bevásárláshoz.
Segítsünk Csillának!

Készítsünk programot, amely sorba összead számokat. A program akkor írja ki az eredményt, ha nullát gépelünk be.

A feladat kicsit összetettebb mint a korábbiak, de az első lépés itt is a feladat elemzése.
Milyen változókra van szükségünk?

szám
összeg

Ha végiggondoltuk, és elkészült a feladatterv, írjuk meg a programot!

Feladatterv:

10 Valtozok: szam, osszeg, nul.
15 Legyen nul = 0.
20 Legyen az osszeg = 0.
30 Kerd be a szamot.
40 Add szamot osszeghez ,tedd osszegbe.
50 Ha osszeg = nul ,akkor menj 30-ra.
60 Ird ki az osszeget.
70 Vege.

Ez a program se sokkal bonyolultabb mint az előzőek.

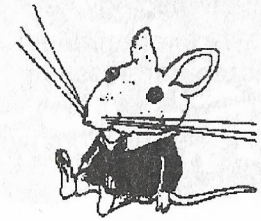
* Keressünk olyan problémákat, melyek megoldásához döntésre van szükség. A legjobb ötletet oldjuk meg együtt!

Feladatterv:

Gyakorló feladatok a W-nyelvű foglalkozásokhoz

* 1.Készíts programot, mely a decimétert átváltja centiméterbe!
* 2.Készíts programot, mely a decimétert átváltja milliméterbe!
* 3.Készíts programot, mely a centimétert átváltja deciméterbe!
* 4.Készíts programot, mely a millimétert átváltja deciméterbe!
* 5.Készíts programot, mely a centimétert átváltja milliméterbe!
* 6.Készíts programot, mely a millimétert átváltja centiméterbe!
* 7.Készíts programot, mely a métert átváltja deciméterbe!
* 8.Készíts programot, mely a métert átváltja centiméterre!
* 9.Készíts programot, mely a métert átváltja milliméterbe!
* 1O.Készíts programot, mely a decimétert átváltja méterre!
* 11.Készíts programot, mely a centimétert átváltja méterbe!
* 12.Készíts programot, mely a millimétert átváltja méterre!
* 13.Készíts programot, mely a dekagrammot átváltja kilogrammra!
* 14.Készíts programot, mely a grammot átváltja kilogrammra!
* 15.Készíts programot, mely a kilogrammot átváltja dekagrammra!
* 16.Készíts programot, mely a kilogrammot átváltja tonnára!
* 17.Készíts programot, mely a dekagrammot átváltja tonnára!
* 18.Készíts programot, mely a tonnát átváltja kilogrammra!
* 19.Készíts programot, mely a tonnát átváltja dekagrammra!
* 2O.Készíts programot, mely a kilogrammot átváltja dekagrammra, grammra!
* 21.Készíts programot, mely a kilogrammot átváltja grammra!

* 22.Készíts programot, mely a grammot átváltja dekagrammra!
* 23.Készíts programot, mely a dekagrammot átváltja grammra!
* 24.Készíts programot, mely a dekagrammot átváltja kilogrammra, grammra!
* 25.Készíts programot, mely tetszőleges másodpercet átvált percre, órára!
* 26.Készíts programot, mely tetszőleges percet átvált órára, másodpercre!
* 27.Készíts programot, mely a napokat átváltja órára!
* 28.Készíts programot, mely a napokat átváltja percekre!
* 29.Készíts programot, mely a napokat átváltja másodpercekre!
* 3O.Készíts programot, mely a napokat átváltja óra, perc, másodpercre!
* 31.Készíts programot, mely tetszőleges órát átvált percre!
* 32.Készíts programot, mely tetszőleges órát átvált másodpercre!
* 33.Készíts programot, mely tetszőleges percet átvált másodpercre!
* 34.Készíts programot, mely tetszőleges órát átvált percre, másodpercre!
* 35.Készíts programot, mely tetszőleges másodpercet átvált percre!
* 36.Készíts programot, mely tetszőleges percet átvált órára!



Csipet-csapat feladatok

1, Csip és Dél szereti a mogyorót. Elmentek az erdőbe, hogy gyűjtsenek maguknak. Csip 35 darabot, Dél 47 darab mogyorót gyüjtött!
   a, Hány mogyorót gyűjtöttek összesen?
   b, Hány darabbal gyűjtött többet Dél?
   Készíts megoldási tervet, és Word nyelvű programot!

2, Sziporka húsvétra tojásokat festett. Összesen 34-et. Ez 8-cal több, mint amennyit az előző évben festett! Hány tojást festett az előző évben? Készíts feladattervet és Word nyelvű programot!
   Készíts megoldási tervet, és Word nyelvű programot!

3, Sziporka 65 szem cukrot kapott egy találmányáért! Elhatározta, hogy a csapattagok között igazságosan szétosztja (5 csapattag).
   a, Hány szem cukrot kapott Csip?
   b, Hány szem cukrot kapott a két mókus összesen?
   Készíts megoldási tervet, és Word nyelvű programot!

4, Sziporka színházba vitte barátait. Egy jegy 7 mogyoróba került. Hány mogyorót fizetett az 5 jegyért?
   Készíts megoldási tervet, és Word nyelvű programot!

5, Kvad Lipi szereti a sajtot. Naponta 9 szeletet fogyaszt el.
   a, Hány szelet sajtot fogyaszt el egy hét alatt?
   b, Hány szelet sajtot fogyaszt el márciusban?
   c, Hány szelet sajtot fogyaszt el egy év alatt?
   Készíts megoldási tervet, és Word nyelvű programot!

6, Lipi egy hatalmas kerek sajtot kapott. Felszeletelte és a hűtőbe tette. Naponta 5 szeletet evett meg. Igy a sajt 3 hétig volt elég.
   a, Hány szeletre vágta fel Lipi a sajtot?
   b, Hány szeletet evett meg egy hét alatt?
   c, Hány egész sajt kellene Lipinek ebédre, ha 3 hónapig szeretne ilyen szép kerek sajtot ebédelni? (Egy hónapot 4 héttel számolj!)
   Készíts megoldási tervet, és Word nyelvű programot!

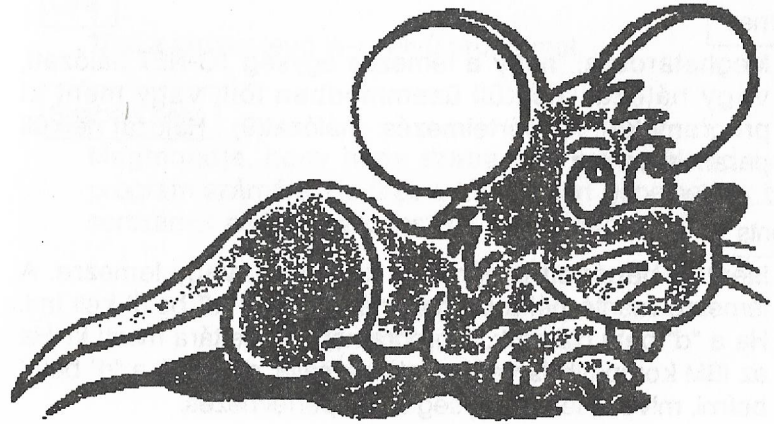
7, A Csipet-csapat moziba ment. Ebben a moziban 200 hely van. A fele és még 20 a nézők száma.
   a, Hány üres hely van a moziban?
   b, Hány jegy kelt el az előadásra?
   Készíts feladattervet és Word nyelvű programot!

Mesés feladatok

8, A törpék a bányában drágaköveket bányásznak. Egy nap alatt 21 mázsa követ mozgatnak meg. Ha mindenki ugyanannyit mozgat meg, egy törpe hány kilógramm követ mozdít meg?
   Készíts megoldási tervet és Word nyelvű programot.

9, Hamupipőkének a gonosz mostoha kétszer annyi és még 156 szemmel több lencsét szórt a hamuba, mint amennyi borsót. Összesen 1662 szem borsót talált a hamuban!
   a, Mennyi a lencsék száma?
   b, Mennyi a lencsék és a borsók száma összesen?
   c, Hány perc alatt válogatta ki körülbelül Hamupiőke, ha minden szemet egy másodpercig keresett? (Mennyit késhetett így a bálról?)

10, Többsincs királyfi vándorlása során 4 holdtöltét ment, mire megtalálta a királykisasszonyt?
   a, Hány napig tartott az út?
   b, Hány kilométert tett meg, ha naponta 14 órát gyalogolt, és óránként 3 kilométert haladt a célja felé?



A Word programnyelv parancsai

A Word programnyelv parancsainak formai (szintaktikai) szabályai:

- *A parancsok mindíg nagybetűvel kezdődnek*
- *A parancs végére nem teszünk pontot*

Fordits

Lefordítja a begépelt, vagy háttértárról betöltött W-nyelvű programot. Minden program csak a fordítás után indítható el (futtatható). A fordítást csak egyszer kell végrehajtani, ha a programon nem módosítunk.

Fuss

Elindítja (végrehajtja) a lefordított programot.

Ird ki, Ird

Egy változó tartalmának kiíratása. Csak akkor van értelme, ha előtte a Word -nyelvű programot lefordítottuk, és lefuttattuk.

Lista

Kiírja a képernyőre a W-nyelvű programot.

Lemez

Meghatározza, hogy a lemezes egység TC-NET hálózati, vagy hálózat nélküli üzemmódban tölt, vagy ment ki programokat. Alapértelmezés: hálózat(9). Hálózat nélküli parancs: "lemez 8"

Ments

Menti a begépelt programot kazettára, vagy lemezre. A lemezre mentésnél a programnév elé egy "d" betűt kell írni. Ha a "d" betű hiányzik, a programot a kazettára menti ki. Ha az IBM kompatibilis változatot használja, nem kell a "d" betűt beírni, mivel a lemezegység az alapértelmezés.

Tölts

Kazettáról, vagy lemezről betölti a megadott nevű programot. A programnév elé egy "d" betűt írunk, a betöltés a lemezes egységről történik, egyébként a kazettáról tölti be a programot. Ha az IBM kompatibilis változatot használjuk, nem kell a "d" betűt beírni, mivel a lemezegység alapértelmezés.

Katalogus

A parancs a lemezkatalógust írja ki a képernyőre. Csak akkor van értelme, ha van lemezes egység a géphez kapcsolva.

Szamold ki, Szamold, ?

A kulcsszó után írt matematikai kifejezést számolja ki. Az eredményt kiírja a képernyőre.

Valtozok

A parancs kiírja, milyen nevű változókat hoztunk eddig létre, vagy használ a program és ezeknek az értékét. A program változóinak kiíratása csak akkor hatásos, ha a program legalább egyszer lefutott. A parancs jól használható a programhibák felderítésére.

Ujra

Törli a tárban levő W-nyelvű programot.

Szabad hely

Megmondja, hogy hány szabad sor van a Word nyelvű program számára. Ha üres a tár, 5O sort gépelhetünk be! (A sorszámot ne keverjük össze a sorok számával!)

A Word programnyelv utasításai

Az utasítások formai (szintaktikai) szabályai:

- *Minden sor egy-egy mondat.*
- *A mondat 78 jelnél hosszabb nem lehet. (Célszerű, ha a változó neveket rövidítjük)*
- *Az utasítás mindig sorszámmal kezdődik*
- *A sorszám mögötti mondat mindíg nagy betűvel kezdődik.*
- *A mondat végét mindig ponttal (.) zárjuk.*
- *Minden Word nyelvű program, a programban használt változónevek (fiókok) felsorolásával kezdődik.*
- *A begépelt, vagy betöltött programokat futtatás előtt le kell fordítani. A program csak ez után működőképes.*

Programkezdés

Változók:

Az utasítás mögé vesszővel (,) elválasztva felsoroljuk az összes változót (fiókot), amit a programban használunk. Ha a felsorolás nem fér el egy sorban, új sorban a "Valtozok:" kulcsszó után folytathatjuk a felsorolást. (A változó név célszerű, ha nem kötőszó pl.:a, az) Mindíg az adott feladat határozza meg, melyik utasítást célszerű a programban alkalmazni.

Értékadás

A Word nyelvben egy-egy változónak kétféleképpen adhatunk értéket:

- Vagy a programból (a programozó határozza meg egy-egy váltazó tartalmát).
- Vagy a program használójától kérjük be a szükséges adatokat.

Legyen

A mögé írt változónak adhatunk értéket programból. Az utasítást akkor célszerű használni, ha olyan adataink vannak, amelyek tartalma a programozó számára ismertek, és nincs szükség ezek módosítására.
Pl.: Legyen valtozo = 10.

Kerd be, Kerd

Az utasítás segítségével a program felhasználója adhat értéket egy változónak.
Pl.: Kerd be a kilot.

Műveletvégzés

Add ossze, Add

A kulcsszó mondatkezdés. A mögé írt két változó tartalmát összeadja. Az eredményt a "tedd" kulcsszó mögé írt változóba helyezi el.
Pl.: 40 Add elsot a masodikhoz ,tedd eredm. vagy:
40 Add elsot a masodikhoz ,tedd elsobe.

Vond ki, Vond

A kulcsszó mögé írt első változó tartalmából elveszi a második változó tartalmát. Az eredmény az "Add" utasítással megegyező módon áll elő.

Szorozd

A kulcsszó mögé írt két változó tartalmát összeszorozza. Az eredmény az "Add" utasítással megegyező módon áll elő.

Oszd

A kulcsszó mögé írt első változó tartalmát osztja a második változó tartalmával. Az eredményt az "Add" utasítással megegyező módon képzi.

Eredmények, változók kiíratása

Ird ki, Ird

Az utasítás mögé írt érvényes változó tartalmát kiírja a képernyőre. Egy változó akkor érvényes, ha a "Valtozok:" kulcsszó utáni felsorolásban szerepel.
Pl.: 60 Ird ki az eredmenyt.

Ugrás a programban

Menj

Az utasítás hatására, a program az utasítás után írt sorszámmal folytatódik. A sorszámnak érvényes sorszámnak kell lennie!
Pl.: 70 Menj 30.

Döntés a programban

Ha

A mondatkezdő utasítás segítségével egy feltételtől függő elágazás hozható létre. A feltételek:

kisebb <
nagyobb >
egyenlő =

Maga a feltétel megfogalmazása nemcsak ezeket a jeleket tartalmazza. Mindíg hasonlítunk valamit valamihez. (Pl.: a változók tartalmát egy másik változó tartalmához.)

Ha az állítás igaz, a program egy megadott sorszámra ugrik.
Pl.: 50 Ha penz < fizet ,akkor menj 40. vagy:
50 Ha penz < fizet ,menj 40.

Az állításnak mindig egyértelműnek kell lennie! Ennek pontos megfogalmazása a programozó feladata.

Program zárása

Vege

Minden programot ezzel az utasítással kell zárni. Az utasítás végét ponttal kell zárni! A programnak csak egy vége lehet.

A Word programnyelv hibaüzenetei

Hibás parancs

A parancs végére pontot tettünk, vagy olyan szót gépeltünk be, amit nem ismer a tolmácsprogramunk.

Hibás utasitás

Olyan kulcsszót gépeltünk be a programba, melyet nem ismer a tolmács.

Hibás adat

A "Kerd be" utasításnál hibásan gépeltük be a számot.

Hibás a zárójelezés

A nyitó és záró zárójelek száma nem egyezik meg.

Nullával nem lehet osztani

Az üzenet önmagáért beszél! Osztási műveletnél fordul elő ez a hiba.

Ismeretlen változó

Olyan változóra hivatkozunk, amely nem szerepel a "Valtozok:" kulcsszó utáni felsorolásban.

A mondatot nagybetűvel kell kezdeni

A W-nyelvű programsorok nagy betűvel kezdődnek! Minden programsor egy-egy mondat, melyek végét ponttal kell zárni!

Nincs több hely a programnak

Az üzenet azt jelenti, hogy 50 fizikai sornál hosszabb programot akarunk begépelni tolmácsunknak, mely maximum 50 soros programig használható. Ne keverjük össze a sorok számát a sorok sorszámával!

Hiányzó szóköz

A mondaton belül az egyes szavakat szóközzel kell elválasztani!

Nincs program

Fordítani, vagy futtatni próbáltunk tolmácsunkkal, de a memóriában nincs W-nyelvű program.

Kétszer meghatározott változó

A "Valtozok:" kulcsszó után kétszer szerepel azonos nevű változó, tolmácsunk így nem tudja megkülönböztetni, mikor, melyik változót kell használnia.

Túl sok változó

A programban 20-nál több változót próbáltunk meg használni. Tolmácsunk, csak 20 változót tud megkülönböztetni!

Nincs vége utasítás

Minden programot a "Vege" utasítással kell zárni.

Hiányzó vessző

A "tedd", "menj", "akkor" kulcsszavak elől hiányzik a vessző (,).

Hibás szám

A "Legyen" értékadó utasításkor hibásan gépeltük be a számot.

Hiányzó szám

A "Legyen" értékadó utasításnál nem adtunk a változónak értéket.

Nem létező sorra történő hivatkozás

A "Menj", "akkor menj" utasítások mögé olyan sorszámot írtunk, ami nem létezik.

Túl sok konstans

A program túl sok "Legyen" utasítást használ.

Nincs lefordított program

Megpróbáltunk egy olyan programot futtatni, ami nincs lefordítva.

Túl hosszú programnév

A "Ments" parancs mögé túl hosszú programnevet írtunk, így a tolmács ezt a parancsot nem hajtja végre. Rövidebb programnevet kell begépelni, hogy a mentés sikeres legyen.

Túl hosszú a kifejezés

A "Szamold ki" parancs mögé túl hosszú műveletet írtunk.

Hibás kifejezés

A "Szamold ki" parancsnál hibásan gépeltük be a matematikai műveletet.

Tartalomjegyzék

PROGRAMJAIM:

Eddig megjelent kiadványaink

A jegyzetek a számítástechnikát szakosított (tagozatos) oktatási formában tanuló diákok jegyzetei. Az oktatási programhoz tartozó jegyzetek moduláris felépítésüek. Így bármely tantárgyhoz felhasználhatók, ahol a számítógép kezelésének oktatása a cél. (Pl.: a technika tantárgy)

- A C+4 üzembe helyezése és alapvető parancsai I.
- A C+4 üzembe helyezése és alapvető parancsai II.
  (A jegyzetekben leírtak bármely COMMODORE gépre érvényesek)
- A Videoton TV-computer üzembe helyezése és alapvető parancsai I.
- A Videoton TV-computer üzembe helyezése és alapvető parancsai II.
- Az IBM PC üzembe helyezése és a DOS alapfokon I. + mágneslemez
- Az IBM PC és a DOS II. + mágneslemez
- A WORD programnyelv

  Első lépések a programkészítés felé. A programnyelv C64, C128, C+4, TVC és IBM-PC gépeken egységes utasításkészlettel és formai megjelenéssel díjtalanul megkapható a kiadványban található megrendelőért. A programnyelv magyar nyelvű, mondatszerkezetű. A programnyelv a tanulóknak átadható.
- Teknőc grafika LOGO 2.9

  Látványos grafika. A magyar nyelvű LOGO 2.9-es programnyelv, beépített segítségnyújtással. C64, C128, C+4, TVC és IBM-PC gépeken egységes utasításkészlettel és formai megjelenéssel. A programnyelv a tanulóknak átadható.
- A számítástechnika története és a C+4 parancsai III.
- A számítástechnika története és a TV-computer parancsai III.
- A számítástechnika története és az IBM PC DOS parancsok III.
- Basic algoritmusok BASIC nyelven minden gépen I.

További jegyzeteink:

- Microsoft MS-DOS 5.0 alapok /gimnáziumi jegyzet/
- Kémiai számítások /7. osztály/
- Kémiai számítások /8. osztály/

Tervezett kiadványaink a korábbi jegyzetekre alapozva:

- Az IBM-PC és a DOS IV.
- Basic algoritmusok BASIC nyelven minden gépen II.

Minden géptipusra vannak oktatóprogramjaink, tanítási órákhoz, felzárkóztatáshoz, egyéni és napközis foglalkozásokhoz. Kérje az APC-Stúdió **új programkatalógusát**!