Pázmány Péter Katolikus Egyetem
Információs Technológiai Kar

Bevezetés a programozásba I

1. gyakorlat

Programozási alapismeretek, a PLanG programozási nyelv

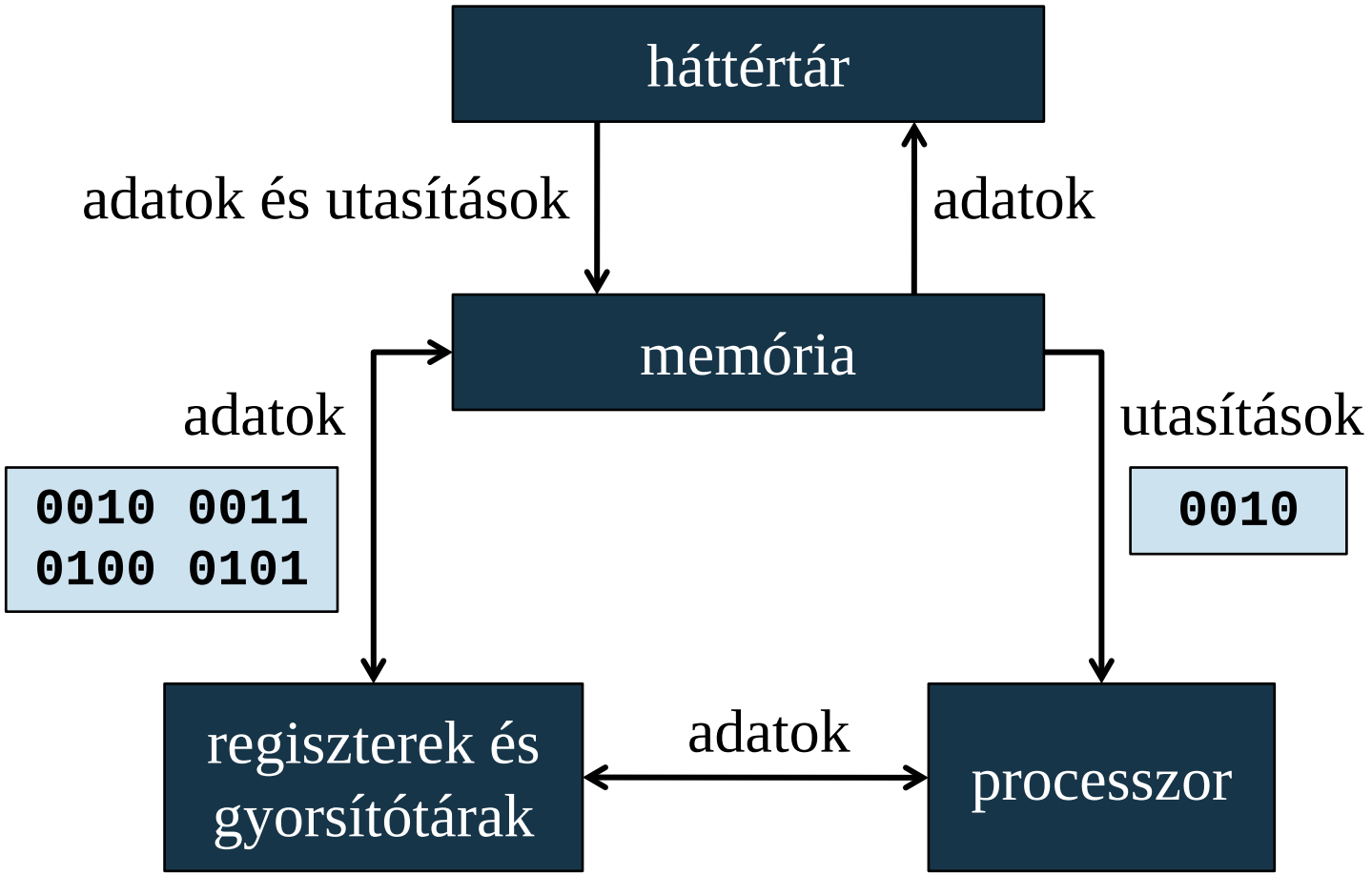
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Programozási alapismeretek

A program

- A *program*
  - *matematikailag*: állapotterek (értékek direktszorzata) felett értelmezett reláció
  - *informatikailag*: utasítások sorozata, amelyek műveleteket hajtanak végre a megadott értékekkel, az *adat*okkal
- A programban foglalt utasítássorozatot, vagy *programkód*ot a *processzor* (CPU, GPU, …) hajtja végre
  - a processzor korlátozott utasításkészlettel rendelkezik, ezért összetett utasításokat nem képes véghezvinni
  - a végrehajtáshoz segédtárakat (regiszterek, gyorsítótárak) használ, és kommunikál a *memória*val
  - az utasítások és adatok binárisan vannak eltárolva

A program futása



Programozási alapismeretek

A programozási nyelv

- A processzor által értelmezhető utasításkészletet és adathalmazt nevezzük *gépi kód*nak (*object code*)
- Mivel a programokat nem tudjuk közvetlenül a processzor feldolgozási szintjén elkészíteni, szükségünk van a működés és az adatkezelés absztrakciójára
- Az absztrakciót megvalósító eszközt nevezzük *programozási nyelv*nek
  - egy adott programozási nyelven megírt programkódot nevezünk a program *forráskód*jának (*source code*)
  - a programozási nyelv meghatározza a használható típusok és utasítások halmazát, amely egy adott nyelvre rögzített, ám a programozó által általában kiterjeszthető

- A programozási nyelvek osztályozása:
  - *alacsony szintű* (assembly): a gépi kódot egyszerűsíti szövegszerűre, de nem biztosít utasításabsztrakciót, pl.:

```
data segment ; adatok
    number dw -5 ; változó létrehozása
data ends
code segment ; utasítások
    …
    mov ax, number ; regiszterbe helyezése
    cmp ax, 0 ; regiszterérték összehasonlítása
    jge label1 ; ugrás, amennyiben nem negatív
    mov cx, 0
    sub cx, ax ; pozitívvá alakítás kivonással
    …
```

- *magas szintű*: a gépi architektúrától független utasításkészlettel rendelkezik, tovább egyszerűsíti az assembly kódot, és további lehetőségeket biztosít a programozó számára, pl.:

```
int main(){ // főprogram
   int number = -5; // változó létrehozása
   if (number < 0) // ha negatív
      number = -number; // ellentettre váltás
   …
}
```

- Magas szinten programozni sokkal kényelmesebb, ugyanakkor általában kevésbé hatékony kódot eredményez

Programozási alapismeretek

Programok fordítása

- A programkód átalakítása rendszerint több lépésben történik, magasabb szintű nyelv esetén először alacsonyabb szintű kód, majd abból gépi kód készül, erre a feladatra szolgál a *fordítóprogram* (*compiler*), magát az átalakítást pedig *fordítás*nak (*compiling*) nevezzük

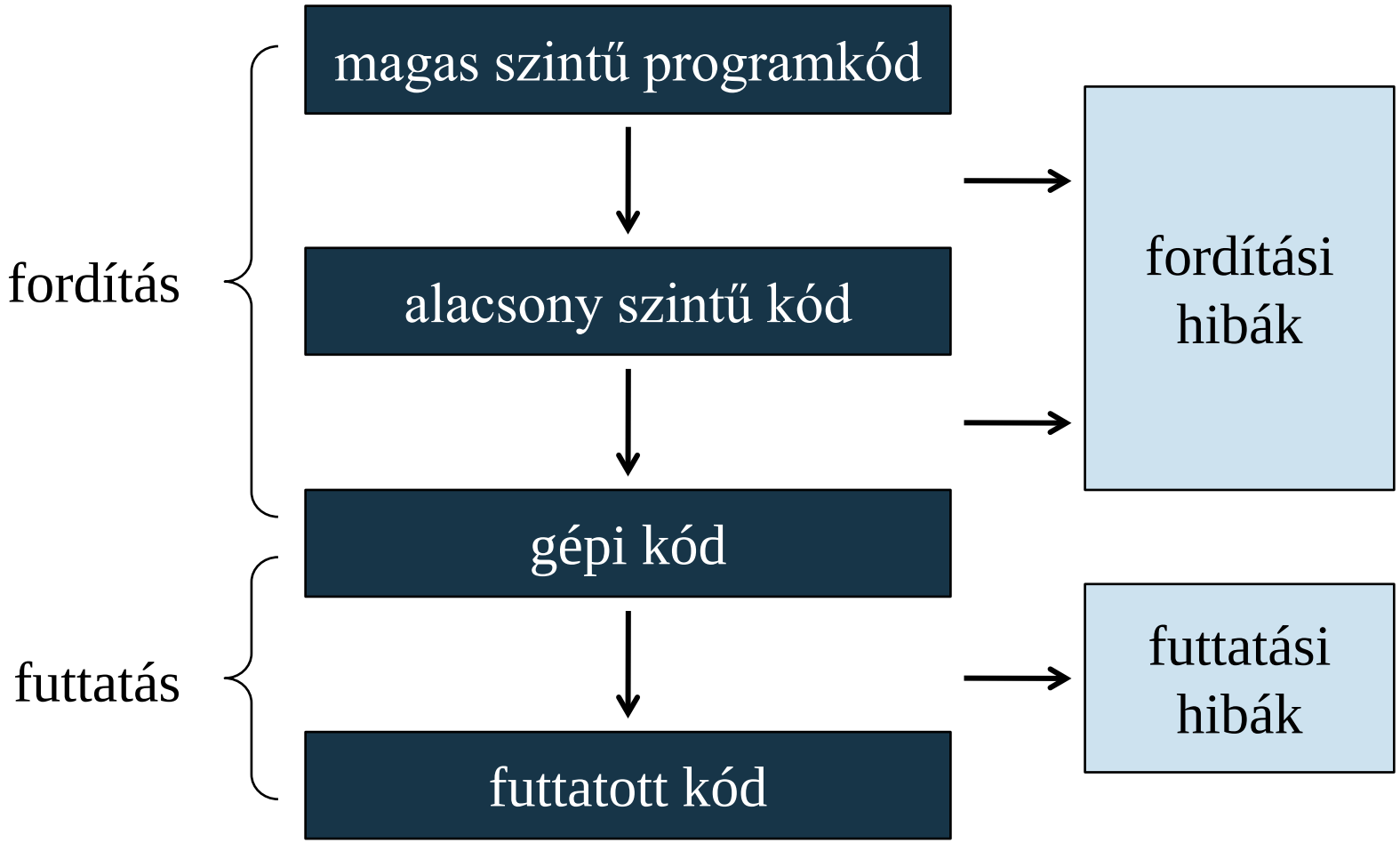
Programhibák

- A programkód tartalmazhat hibákat, amiket a fordító megtalál, amelyeket két kategóriába sorolunk:
  - *szintaktikai*, *vagy elemzési* (*syntax error*): a programkód szerkezete helytelen, pl. hibás utasításnév, zárójelezés
  - *szemantikai*, *vagy értelmezési* (*semantic error*): az érték változásával, a műveletek végrehajtásával bekövetkező hibák, pl. 0-val történő osztás, hibás memóriacím
- A szintaktikai hibákat és a szemantikai hibák egy részét a fordítóprogram megtalálja és figyelmeztet rá, ezért ezeket *fordítási hibák*nak (*compile time error*) nevezzük
  - a fordítóprogram jelzi a hiba lehetséges okát, illetve helyét, de mindig ott találja meg a hibát, ahol elkövettük

Programhibák

- A fordítás során fel nem tárt hibák a programfutás során váltódnak ki, ezek a *futtatási hibák* (*runtime error*, *bug*)
- A futási idejű hibák ellenőrzését *tesztelés*sel végezhetjük
  - a *statikus tesztelés* során a programkódot vizsgáljuk át és keressük a lehetséges hibalehetőségeket
  - a *dinamikus tesztelés* során futás közben keressük a hibákat (pl. szélső értékek, vagy nem megfelelő értékek megadásával)
- A szoftverfejlesztői környezetek megadják a *nyomkövetés* (*debug*) lehetőségét, azaz futás közben végigkövethetjük a kódot és a memóriában tárolt értékeket, vagy akár lépésenként hajthatjuk végre a programkódot

Programok fordítása és futtatása

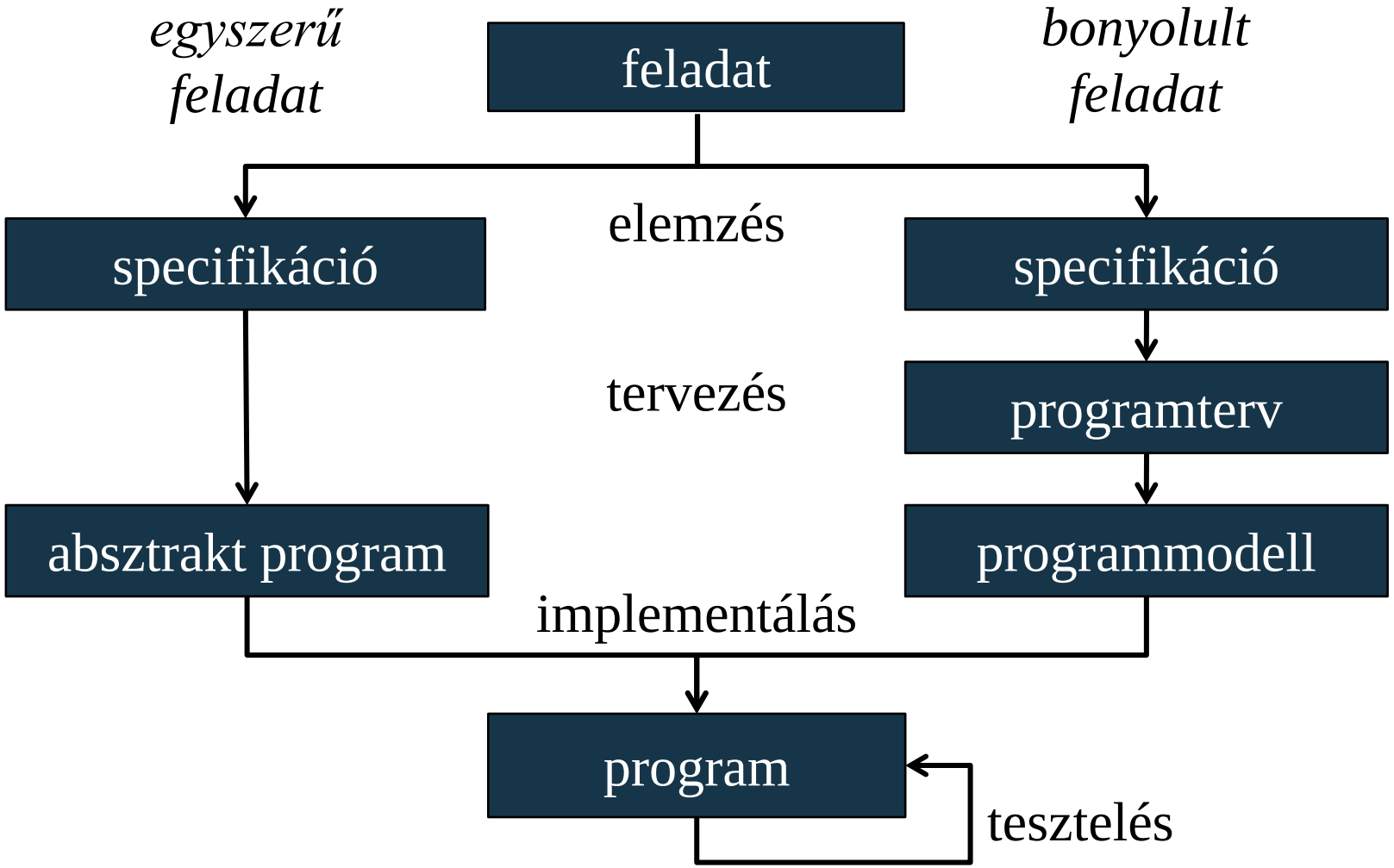


Programozási alapismeretek

A szoftverfejlesztés folyamata

- A szoftverfejlesztés a kódoláson túl több lépésből áll, amely függ a feladat bonyolultságától is:
  1. A feladatot elemezni kell, és megadni a formális megfelelőjét, vagyis a *specifikáció*t
  2. A specifikációt alapján megtervezhető a program, amely egyszerű feladatnál az *absztrakt program*, míg bonyolult feladatnál a *programterv* elkészítésével jár, amelyből elállítható a *programmodell* (egyszerűsített élprogram)
  3. A tervet implementáljuk a megfelelő programozási nyelven
  4. Az implementált programot, illetve a programkódot *tesztelés*nek vetjük alá, ami módosításokat eredményezhet az implementációban (vagy a korábbi fázisokban)

A szoftverfejlesztés folyamata



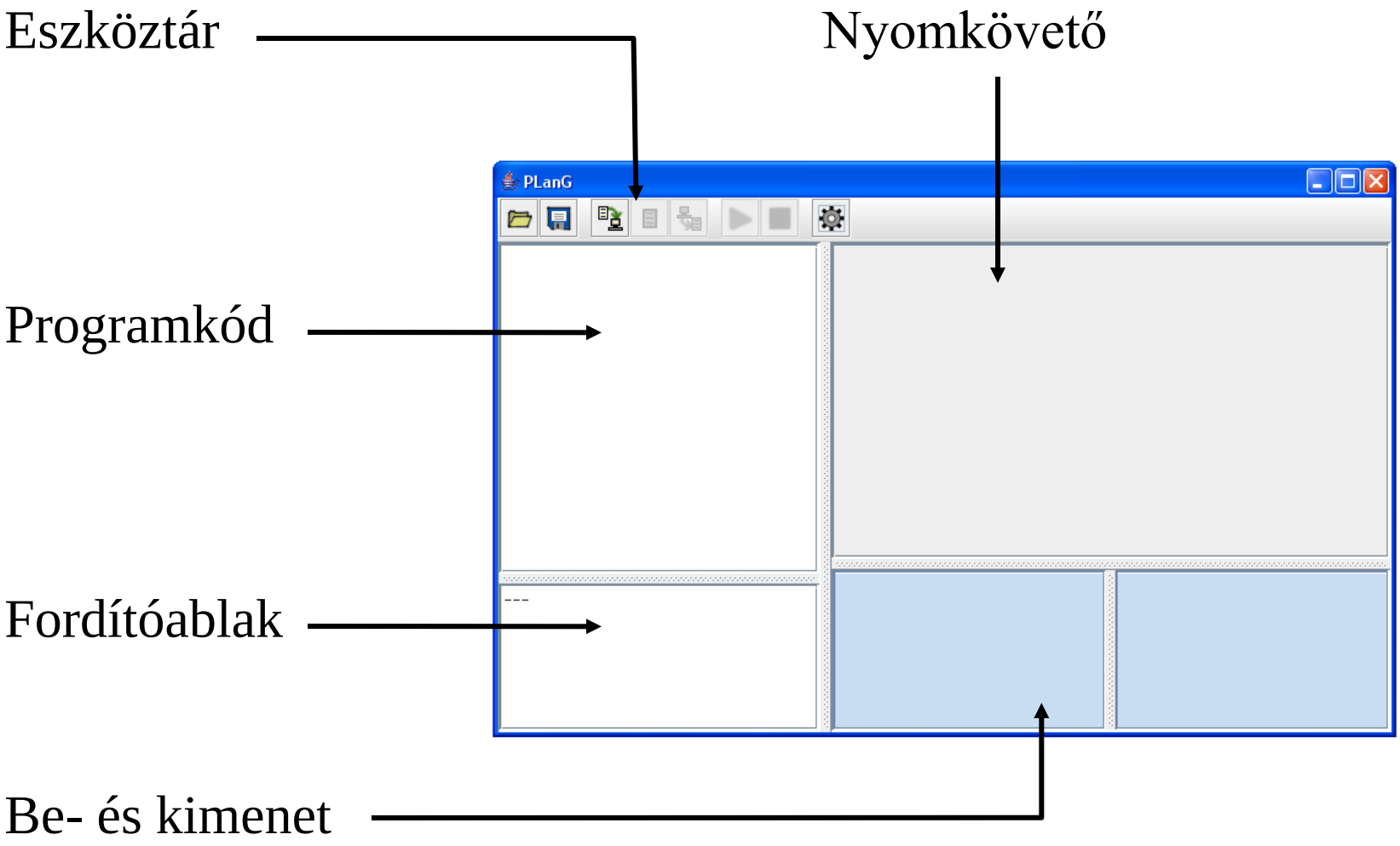
A PLanG programozási nyelv

Eredete

- *Lővei László* fejlesztette ki, célja: programozás oktatása kezdőknek
- *Java*ban íródott, ezért a használatához szükséges a Java futtatási környezet (*Java SE Runtime Environm*ent)
- Egy fordítóprogram, és fejlesztői környezet, amely saját programozási nyelvecskét alkalmaz
- Funkciói: eredmény kiírása, futtatás nyomkövetése lépésenként (azaz megtudhatjuk, egy sor feldolgozása miként történik), memória nyomkövetése lépésenként (láthatjuk változóink értékváltozásait)
- Programozás alapszabályát (CTRL+C/CTRL+V) alkalmazhatjuk

A PLanG programozási nyelv

Működése



A PLanG programozási nyelv

Működése

- *Programkód:* ide gépeljük be az utasításokat a szintakszisnak megfelelően, ez kerül fordításra, majd futtatásra
- *Nyomkövető:* a program lépéseit követhetjük nyomon, itt láthatjuk, változóink milyen értékeket vesznek fel futás közben, és hogyan foglalják a memóriát
- *Fordítóablak:* fordítási üzenetek, illetve, ha hibát talál a programban fordítás során, akkor azok ide kerülnek
- *Bemenet:* beolvasandó értékek, ha a programba szeretnénk magunk megadni értékeket futás közben, valamint itt fognak szerepelni a beolvasott fájlok adatai
- *Kimenet:* kiírás helye, ide kerülnek a program által kiírt értékek, valamint itt fognak szerepelni a fájlba kiírt adatok

A PLanG programozási nyelv

Felépítés

- A programoknak nevet kell adni, amit az első sorban írunk a **PROGRAM** kulcsszó után
- A programok elején a deklarációs rész található, amit a programtörzs követ, a kettő között nem kell semmilyen kulcsszót tenni
- A deklarációs részt a **VÁLTOZÓK** kulcsszó jelöli
- A programtörzs végét a **PROGRAM_VÉGE** kulcsszó jelöli

```
PROGRAM programnév
    VÁLTOZÓK:
          ... ** változók deklarációja
    ... ** programtörzs
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Felépítés

- A program nevében és a változónevekben bármilyen ékezetes, alfanumerikus karaktert használhatunk, de csak betűvel kezdődhet (pl. a120 megfelelő, de 1a20 már nem)
- A nagy és kisbetűket nem különbözteti meg a PLanG (illetve mindig megformázza a szöveget, hogy ne legyenek ilyen problémák)
- Megjegyzéseket (** után) bármelyik sor végén, vagy külön sorban elhelyezhetünk, és bármit írhatunk a sor végéig
- Szöveget idézőjelben (") adunk meg, ebben bármilyen karakter szerepelhet, a szöveg pedig bármennyi karaktert tartalmazhat
- Karaktert szimpla idézőjelben (') adhatunk meg
- Több szöveget is konkatenálhatunk, vessző segítségével

A PLanG programozási nyelv

Példa

Feladat: Írjuk ki a „Hello, World!” feliratot a kimenetre.

- a kiíratásnál megadjuk a szöveget, amit ki szeretnénk íratni
- a programban ezen kívül más sorra nincs szükség
- mivel nincs változók a programban, ezért a deklarációs részt kihagyjuk, csak a programtörzs rész kell
- adjuk a programnak a **`hello_world`** nevet

Specifikáció:

- bemenet: nincs
- kimenet: „Hello, World!” felirat

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM hello_world
  ** a program megkapta a hello_World nevet
  KI: "Hello, World!"
  ** ez a lényegi rész, amikor ez a sor lefut,
  ** megtörténik a kiírás, nem kell várni a
  ** program végére
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

- Másféle megoldás is adható, amely ugyanazt az eredményt adja:

```
PROGRAM hello_world
  KI: "Hello,"
  KI: " World!"
PROGRAM_VÉGE
```

  - a kiíratást két sorban adtuk meg, de mivel nem írtunk sortörést, egy sorba kerül az eredmény
  - általában igaz, hogy egy problémára sokféle program adható, minél bonyolultabb a feladat, annál többféle

A PLanG programozási nyelv

Fordítás

- A programkódot fordítjuk, majd futtatjuk
  - először értelmezzük a programunkat, ha jól írtuk be, a fordító nem ad hibát
  - futtatással megkapjuk az eredményt a kimeneti ablakban, a fordítási ablakban „a program véget ért” üzenettel
  - a nyomkövető ablakban megjelennek a lépések, minden sor végrehajtása egy lépés (a deklaráció nem tartozik bele), ezeken lépkedhetünk is, a fordítóablakban megkapjuk az aktuális lépést
- Ha hibásan írjuk meg a programot, természetesen nem futtathatjuk, a hiba valószínűsíthető helyét látjuk, váltsunk vissza szerkesztő módba, majd javítsuk a hibát

A PLanG programozási nyelv

Kifejezések

- A program törzsrészében az értékekkel műveleteket végezhetünk (összeadás, és, szinusz, …), amelyek eredménye egy újabb érték lesz, az ilyen műveleteket nevezzük *kifejezések*nek
- Amikor egy kifejezés lefut, és keletkezik az új érték, azt a kifejezés *kiértékelés*ének nevezzük
- Pl.:
  - **`1 + 8`** (a bemenet két egész szám, az eredmény egy egész szám)
  - **`X vagy igen`** (ha **`x`** egy logikai változó, vagy konstans, az eredmény egy logikai élrék)
  - **`3 >= 1`** (logikai értéket ad, amely **`igaz`** lesz)

A PLanG programozási nyelv

Példa

Feladat: Írassuk ki a 124 + 123 összegét.

- a kimenetre egy kifejezést adunk, amelye a program elvégez a kiírás előtt

Specifikáció:

- bemenet: nincs
- kimenet: 247

Megoldás:

```
PROGRAM osszeg
  KI: 124 + 123
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Típusok

- A PLanG-ban (ahogy a legtöbb programozási nyelvben) az értékek (változók, konstansok) típussal rendelkeznek:
- **`LOGIKAI`**: olyan változó, amely logikai értéket reprezentál, IGAZ és HAMIS értéket vehet fel, és logikai műveletek értelmezhetőek rajta (ÉS, VAGY, NEM), illetve az egyenlőségvizsgálatok logikai eredményt adnak
- **`EGÉSZ`**: egész szám a következő műveletekkel: összeadás, kivonás, szorzás, osztás, egész osztás, maradékképzés, negáció, abszolút érték, hatványozás
- **`VALÓS`**: valós (lebegőpontosan ábrázolt) szám, az előbbiek mellett a matematikai függvények, az egészrész képzés, és a kerekítés értelmezhetőek rajta, de a maradékképzés és az egész osztás nem

A PLanG programozási nyelv

Típusok

- **KARAKTER**: betűk, számjegyek, írásjelek, szóköz és sorvége jel (SV), utóbbi kivételével szimpla idézőjelben, amelyet lehet nagybetűvé, és kisbetűvé konvertálni, illetve lekérdezni, hogy szám-e, vagy betű-e
- **SZÖVEG**: több karakter egymásutánja dupla idézőjelben, lehet összefűzni, szövegrészt lekérdezni, illetve karaktert keresni és lekérdezni a szövegből
- **FÁJL**: külön megadhatunk kimeneti és bemeneti fájlokat, amelyeket megnyithatunk, olvashatunk, írhatunk, és bezárhatunk, továbbá lekérdezhetjük, hogy végére értünk-e a fájlnak

A PLanG programozási nyelv

Példa

Feladat: adjuk meg egy egész szám rákövetkezőjét

- be kell olvasnunk egy számot a bemenetről
- meg kell növelnünk az értékét eggyel
- ki kell íratnunk a megnövelt értéket
- szükségünk lesz egy változóra, amibe a bemenő értéket eltároljuk, legyen a neve: a, a típusa: egész
- a beolvasandó értéket a program indítása előtt meg kell adni a bemeneti ablakban

Specifikáció:

- bemenet: egy egész szám (a)
- kimenet: a szám rákövetkezője (a + 1)

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM rakovetkezo
  VÁLTOZÓK:
    a: EGÉSZ
  BE: a
  KI: a + 1
PROGRAM_VÉGE
```

- Ha hibás típusú értéket teszünk be (pl. egész helyett valósat, vagy szöveget), vagy nem írunk be értéket a program futtatása előtt, akkor *futási hibát* kapunk
- Amikor megnöveljük az értékét, előbb végrehajtódik az összeadás, és a kapott érték kerül a kimenetre

A PLanG programozási nyelv

Példa

- Szöveget is írhatunk a kiíratáshoz, hogy látványosabb legyen, akkor a kiíráskor összeillesztjük a szöveget a számmal:

```
PROGRAM rakovetkezo_kiirassal
  VÁLTOZÓK:
      a: EGÉSZ
  BE: a
  KI: "A rákövetkezője: ", a + 1
  ** itt összekonkatenáljuk a kiírandó dolgokat
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Értékadás

- Használhatunk egy másik változót (legyen ez b), amibe előbb betesszük az új értéket, majd kiíratjuk azt
- Ehhez *értékadás*t (:=) kell használnunk,
  - az értékadásnak van bal oldala, és jobb oldala
  - az ott található értékeket nevezzük balértéknek, illetve jobbértéknek
  - balérték csak változó lehet
  - jobbérték tetszőleges kifejezés lehet
- A változókat vesszővel választjuk el egymástól, ha ugyanolyan típusúak, nem kell mindegyiknek megadni a típusát, csak az utolsónak (ekkor az összes előtte lévő ugyanolyan típusú lesz)

A PLanG programozási nyelv

Példa

Specifikáció:

- bemenet: egy egész szám (a)
- kimenet: a b egész szám eggyel nagyobb az a-nál

Megoldás:

```
PROGRAM rakovetkezo_ertekadassal
  VÁLTOZÓK:
     a, b: EGÉSZ
  BE: a
  b := a + 1 ** a b változóba a jobb oldali
             ** kifejezés értékét tesszük
  KI: a, " rákövetkezője: ", b
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

- Értékadással növelhetjük ugyanannak a változó értékét kiíratás előtt, és ezt követően íratjuk ki az új értéket
- Ekkor a változó egyszerre lesz jobb érték, illetve bal érték az értékadásnál

```
PROGRAM rakovetkezo_egy_valtozoval
  VÁLTOZÓK:
      a: EGÉSZ
  BE: a
  a := a + 1
  ** a jobb oldalon a régi érték van, a balon
  ** az új érték lesz
  KI: "A rákövetkezője: ", a
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

Feladat: Adjuk meg egy szám szinuszát.

- a nyelvben megtalálható a **`SIN`** utasítás, amely egy valós, vagy egész számot tud fogadni, és megadja annak szinuszát, valós számként
- a feladat nem határozta meg, milyen számot vegyünk, ezért azt a specifikáció során pontosítjuk
- A művelet által kapott eredményt közvetlenül konkatenálhatjuk egy szöveggel és kiírhatjuk

Specifikáció:

- bement: egy valós szám (sz)
- kimenet: a szám szinusza

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM szinusz
    VÁLTOZÓK:
       sz: VALÓS
    BE: sz
    KI: "A szám szinusza: ", SIN sz
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

Feladat: Írjuk ki, hogy a beolvasott karakter szám-e.

- a karakterekre értelmezett SZÁM művelet visszaad egy logikai értéket, hogy az számjegy-e
- egy teljes szövegre ne,m alkalmazható, azaz ha egy szöveget kéne megvizsgálni, akkor azt karakterenként kell végignézni

Specifikáció:

- bement: egy karakter (char)
- kimenet: igaz, ha a karakter szám, különben hamis

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM betu_e
    VÁLTOZÓK:
       char: KARAKTER
    BE: char
    KI: SZÁM char ** eredménye logikai érték lesz
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

Feladat: Olvassunk be egy egész és egy valós számot, és írjuk ki a hányadosukat.

- egy egész és egy valós számot olvasunk be egymás után, az első számot osztjuk a másodikkal
- a beolvasásnál egyszerre két értéket olvasunk be, azokat vesszővel választjuk el a kódban, szóközzel a bemeneten
- egy harmadik értékbe írjuk az eredményt, amelyre használjunk valós változót

Specifikáció:

- bemenet: egy egész szám (a) és egy valós szám (b)
- kimenet: a két szám hányadosa a harmadik (c) számban

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM hanyados
  VÁLTOZÓK:
     a: EGÉSZ,
     b, c: VALÓS
     ** most már több típusú változónk is van
  BE: a, b
  ** egymás után olvasunk be két számot a
  ** bemenetről
  c := a / b
  KI: "A számok hányadosa: ", c
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

Feladat: Osszunk el két számot maradékosan úgy, hogy a maradékot is megadjuk, és, hogy az első szám több, mint tízszerese-e a másodiknak.

- használjuk a DIV és MOD műveleteket, illetve egy logikai kifejezést
- a hányadosról lekérdezzük, hogy nagyobb-e 10-nél
- az eredmények eltárolására két egész és egy logikai érték

Specifikáció:

- bement: két egész szám (a, b)
- kimenet: a két szám hányadosa (c) és maradéka (d), illetve, hogy a hányados nagyobb-e 10-nél (l)

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM egesz_osztas
  VÁLTOZÓK:
     a,b,c,d: EGÉSZ,
     l: LOGIKAI
  BE: a,b
  c := a DIV b ** egész értékű hányados
  d := a MOD b ** maradék
  l := (c >10) ** logikai értékű kifejezés
  KI: "A hányados: ", c, ", a maradék: ", d,
      " a hányados több, mint tízszerese a
      nevezőnek: ", l
  ** több mindent iratunk ki egyszerre
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Szövegkezelés

- A bemeneti ablakon nem kell külön jelölnünk, milyen adatokat adunk meg, mert attól függően értelmezi a PLanG az értéket, hogy milyen típusú változóba olvasunk be
- Ha értékadás segítségével adunk meg valamit, akkor a következő három megadási lehetőségünk van:
  - jelölés nélkül: (egész, valós) számok, sorvége jel, pl.: 1, 3.123, SV
  - egyszeri idézőjel (**'**): karakterek, pl.: **'k'**, **' '**, **'*'**
  - kétszeri idézőjel (**"**): szöveg, pl.: **"Hello!"**, **" "**, **"k"**, **""**
    - egy karakter is lehet szöveg, ha úgy adjuk meg
    - üres szöveget is megadhatunk, ha nem írunk semmit az idézőjelbe

A PLanG programozási nyelv

Példa

Feladat: Olvassunk be egy szöveget a bementről, és rakjunk a végére egy felkiáltójelet, majd írjuk ki a szöveg hosszát.

- az összeadással tudunk szöveget konkatenálni
- végezzük el a konkatenációt, majd rakjuk az eredményt vissza a változóba
- a hosszt a **|<*szövegnév*>|** jelöléssel tudjuk lekérdezni
- a két kiíratás közé tegyünk sorvége jelet

Specifikáció:

- bemenet: egy szöveg (szo)
- kimenet: a szöveg és egy felkiáltójel, valamint a szöveg hossza

A PLanG programozási nyelv

Példa

Megoldás:

```
PROGRAM felkialtojel
    VÁLTOZÓK:
       szo: SZÖVEG
    BE: szo
    szo := szo + '!'
    KI: szo, SV
    ** kiírja a szót, és sortörést végez
    KI: "A hossza: ", |szo|
    ** a következő sorba a hosszát
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Példa

Feladat: Írassuk ki a bementi szöveg első betűjét.

- többféle megoldás közül is válogathatunk
- csak egy karaktert olvasunk be, és azt írjuk ki, vagy
- lekérdezzük a beolvasott szöveg első karakterét (ez a 0. indexű lesz), egy szöveg tetszőleges karakterét lekérdezhetjük **<*szövegnév*>[<*sorszám*>]** formában, vagy
- lekérdezzük az 0. karakternél kezdődő, 1. karakternél végződő szövegrészt, egy tetszőleges szövegrészt a **<*szövegnév*>[<*kezdő sorszám*>:<*hossz*>]** formában tudunk lekérdezni

A PLanG programozási nyelv

Példa

Specifikáció:

- bemenet: egy szöveg (szo)
- kimenet: a szöveg első karaktere (kar)

Megoldás:

```
PROGRAM elso_betu
    VÁLTOZÓK:
       szo: SZÖVEG, kar: KARAKTER
    ** első megoldás:
    BE: kar
    KI: kar ** csak az 1. karaktert nézzük
```

A PLanG programozási nyelv

Példa

```
    ** második megoldás:
    BE: szo
    kar := szo[0] ** az első karakter
    KI: kar ** karakter lesz az eredmény
    ** harmadik megoldás:
    BE: szo
    szo := szo[0:1] ** szövegrész lekérdezés
    KI: szo ** szöveg lesz az eredmény
PROGRAM_VÉGE
```

A PLanG programozási nyelv

Feladatok

I. Kifejezések:

2. Döntsd el egy tetszőleges számról, hogy egy adott intervallumba esik-e. (Először beolvassuk a számot, azt követően pedig az intervallum minimumát, majd maximumát.)
8. a) Add meg egy számtani sorozat első két elemének ismeretében a harmadik elemét.

   b) Add meg az N-edik elemét.
9. (*) Számítsd ki egy háromszög területét az oldalhosszaiból.
12. (*) Add meg egy másodfokú egyenlet megoldásait. (A másodfokú egyenlet 3 tényezőjét olvassuk be.)
15. Döntsd el egy szövegről, hogy nagybetűvel kezdődik-e.