

# Programozás módszertan

## *Programozási nyelvek*

Pere László (pipas@linux.pte.hu)

PÉCSI TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KAR  
INFORMATIKA ÉS ÁLTALÁNOS TECHNIKA TANSZÉK

PTE TTK SZÁMÍTÁSTECHNIKAI SZOLGÁLTATÓ KÖZPONT

MAGYAR TUDOMÁNYOS AKADÉMIA  
SZÁMÍTÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZET  
ELEARNING OSZTÁLY

# *A programozási nyelvek csoportosítása*

# Imperatív nyelvek

---

A latin *imperare* parancsolni szóból származik a kifejezés. Szokásos elnevezés még a „procedurális” (*procedure*, eljárás).

Az imperatív módszertan lényege, hogy utasítások sorozataként megvalósított programmal a számítógép operatív memóriájának egy részét – a változókat – változtatjuk. A program által kiváltott memóriaállapotok végül a megoldáshoz vezetnek.

# Funkcionális nyelvek

---

A funkcionális módszertan függvények (*function*, függvény) segítségével írja le a probléma megoldását. A teljes program egy kiértékelendő függvény az adott bemeneti értékekre mindig olyan értéket ad, amely a feladat megoldását adja.

A függvények értékeit kizárólag az argumentumok értékei határozzák meg, azaz a függvény nem rendelkezik belső állapotokkal, változókkal. (Az imperatív programok részleteit is szokás függvénynek nevezni, de azoknak belső állapotaik lehetnek.)

# Logikai nyelvek

---

A logikai vagy szabály alapú nyelvek a formális logika, a halmazelmélet eszközeivel egy szabályrendszert írnak le. A program futtatása az adott tényhalmaz (bemeneti adatok) és a szabályrendszer által elvégzett következtetés, bizonyítás.

A programozási nyelv ugyanazt az algoritmust használja minden probléma megoldására.

# Objektumorientált nyelvek

---

Az objektumorientált programok a valóság létező vagy elvont elemeit modellező objektumokból épülnek fel. Az objektumok belső változókkal (állapottal) és végrehajtható programrészekkel (metódusok) rendelkeznek.

Az objektumorientált módszertan a következő slide-okon látható három szakaszon ment át:

# Objektum alapú módszertan

---

A változók és metódusok objektumokká szervezhetők,  
de nincs eszköz az objektumok rendszerbe szervezésére.

# Osztály alapú módszertan

---

Az objektumok osztályokba csoportosulnak, a különféle osztályok közti kapcsolatot azonban nem kezeljük. Az objektum az osztály egy előfordulása, példánya, rendelkezik mindazon változókkal és metódusokkal, amivel az osztály többi objektuma, de a változók értéke, az állapot a sajátja. (Analógia: változótípus és konkrét változó.)



# Objektumorientált módszertan

---

Az osztályok közti viszonyok leírására az öröklődési rendszert használhatjuk.

# Absztrakció szerinti felosztás

---

A legalsó szintet a gépi kód alkotja. Ez a szint nem vonatkoztat el a számítógéptől, közvetlenül az általa végrehajtott utasításokat tartalmazza.

Az alacsony szintű programozási nyelvek kevésbé, a magas szintű nyelvek jobban elvonatkoztatnak a gépi kódtól.

# Cél szerinti felosztás

---

Léteznek általános célú programozási nyelvek és léteznek speciális nyelvek, amelyek adott célra használhatóak vagy adott célra optimalizáltak.

# *Programozási nyelvek*

# ABC

---

Az ABC programozási nyelv a BASIC programozási nyelv fejlettebb változataként a programozói munka vizsgálatával, annak megkönnyítésére készült.

A nyelv támogatja a következő adattípusokat: karakterláncok, tetszőleges pontosságú számok, rekordok, listák és asszociatív tömbök.

Az ABC programozási nyelv segítségével természetes nyelvű interface-ek készültek.

Az ABC a nyelvi szerkezetek jelölése a tördelést (*indentation*) használja.

# ABC

```
HOW TO RETURN index doc:
    PUT {} IN where
    FOR line.no IN keys doc:
        TREAT LINE
    RETURN where
TREAT LINE:
    FOR word IN split doc[line.no]:
        IF word not.in keys where:
            PUT {} IN where[word]
    INSERT line.no IN where[word]
```

# Ada

---

Az Ada programozási nyelvet egy tervezőcsoport hozta létre a *Department of Defense* megrendelésére. A nyelvet LADY ADA AUGUSTA BYRON-ról nevezték el.

A tervezés legfontosabb szempontjai a következők voltak:

- megbízhatóság
- továbbfejleszthetőség
- következetesség
- erős típusosság

Eredetileg a DoD csak az Ada nyelvet fogadta el szoftverfejlesztéseiben, de ez a szigor 1987-ben enyhült.

# Ada

---

Az Ada néhány jellemző szolgáltatása:

- beágyazott alprogramok
- beágyazott csomagok
- erős típusosság
- multi-tasking
- kivételkezelés
- absztrakt adattípusok

Az Ada támogatja a következő adattípusokat: numerikus, logikai, karakter, referencia, felsorolás, tömbök, rekordok, és karakterláncok.



# Ada

---

Az Ada 83 nagyobb programrendszerek fejlesztésére (katonai, kereskedelmi, telekommunikációs) kitűnően bevált.

Az Ada legújabb változata az Ada 95, amely többek közt objektumorientált eszközökkel bővült.

Az Ada 83 és az Ada 95 szintén ISO szabvánnyal szabályozott.

# Ada

```
-- simple programming with floating-point #s
with Ada.Float_Text_IO;
use Ada.Float_Text_IO;
procedure Think is
    A, B : Float := 0.0; -- A and B initially zero; note the period.
    I, J : Integer := 1;
begin
    A := B * 7.0;
    I := J * 3;
    B := Float(I) / A;
    Put(B);
end Think;
```

# Algol

---

Tudományos és ipari számítások elvégzésére készült programozási nyelv, mely a nevét a *algorithmic language* kifejezés rövidítéséről kapta.

Az algol az első nyelv:

- ahol rekurzió használható
- amelyben a programozó adattípusokat hozhat létre
- amelyet BNF (Backus Naur Form) segítségével formálisan is leírtak

A nyelv első változatát egy bizottság tervezte és John Backus implementálta.

# Algol

---

Az Algol60 a következő típusokat támogatta: logikai, egészek, valósak és karakterláncok valamint tömbök.

Az Algol68 jelentősen fejlettebb típusrendszere már összetett típusok létrehozását is támogatta.

Az Algol60 erősen típusos nyelv volt, ami az Algol68-ra még jellemzőbb vonássá vált.

Az Algol nyelv mára gyakorlatilag kihalt, ellentétben például a Fortran néven ma is közkedvelt kortársával, maga a nyelv azonban igen nagy hatással volt a későbbi programozási nyelvekre.

# Algol

```
procedure problem (a, b);  
  value a, b; integer a, b;  
  begin integer k; real e;  
    for k := 2 × (a ÷ 2)    1 step 2 until b do  
      begin  
        e := if prime(k) then sqrt(3 × k    sin(k))  
              else sqrt(4 × k    cos(k));  
        if prime(k) then putlist(k, e, 'prime')  
          else putlist(k, e, 'nonprime')  
      end  
    end problem;
```

# APL

---

Az APL nyelvet az IBM fejlesztette ki az 1960-as években. A nyelv erős aritmetikai és mátrixalgebrai eszközökkel rendelkezik.

Az APL nyelv segítségével a hozzáértő programozó nagyon gyorsan és hatékonyan képes matematikai célokra alkalmazásokat készíteni, de a nyelv szokatlan szerkezete miatt sokáig tart hozzáértő programozóvá válni.

A nyelv bizarr karakterkészletet használ, ami megnehezíti a munkát.

# AppleScript

---

Az AppleScript az Apple Macintosh környezet szkript nyelve, amelynek sok különleges tulajdonsága van:

- a program közbülső kódra fordul futtatás előtt
- a nyelv a beszélt nyelvekhez hasonlóan épül fel
- objektumorientált
- a UNIX rendszerek filozófiáját csempészi a MacOS-be

Az AppleScript a HyperTalk továbbfejlesztett változata.

# AppleScript

---

```
-- Use the Finder to close all applications
-- (by Joshua D. Baer)

property specialApps : {"Finder"}

tell application "Finder"
    set allApps to name of processes
end tell

repeat with someParticularApp in allApps
    if specialApps does not contain someParticularApp then
        tell application someParticularApp
            activate
            quit
        end tell
    end if
end repeat
```



# Autolisp

---

Az Autolisp a Lisp nyelvjárása, amelyet a Autodesk alkalmazások (pl. AutoCAD) vezérlésére használhatunk.

# AWK

---

Az AWK programozási nyelv része a *POSIX Command Language and Utilities standard* szabványnak, így minden szabványos operációs rendszernek támogatnia kell.

Az AWK elsősorban szöveges adatállományok feldolgozására szolgál, igen egyszerűen lehet a segítségével szöveges állományokat soronként feldolgozni.

Az AWK programozási nyelvben használt szemléletet nevezik adatvezérelt (*data driven*) szemléletnek is.

# AWK

---

Az AWK eredetileg saját célra készült, a UNIX felhasználók a kezdeti években magánúton másolták a programot. Ez magyarázza a program különös nevét, amelyet szerzőiről (Aho, Kernighan és Weinberger) kapott.

Különösen jellemző a nyelvre a sorolvasást végző szerkezet, a szabályos kifejezések erőteljes támogatása és a típusrendszer hiánya.

Sokak szerint a Perl napjainkban felváltja az AWK nyelvet.

# BASIC

---

A BASIC nyelvet John Kemény és Thomas Kurtz vezetésével fejlesztették 1964-ben. Nevét a Beginner's All Purpose Symbolic Instruction Code rövidítéseként kapta.

A BASIC IBM 704, majd DEC PDP-11 gépeken vált népszerűvé, köze sem volt hozzá a személyi számítógépeknek és a Microsoftnak.

A nyelvet elsősorban kezdők (nem szakemberek) számára fejlesztették, ennek köszönhetően hamar népszerű lett.

Az eredeti BASIC nyelvre igen jellemző volt, hogy minden programsort sorszámmal kellett kezdeni, amelyek címkeként funkcionáltak.

# BASIC

---

Az eredeti BASIC nyelv numerikus és string, valamint tömb adattípussal rendelkezett.

A BASIC nyelvet a megjelenő újabb és újabb nyelvjárások nagymértékben fejlesztették és a felismerhetetlenségig eltorzították.

# BCPL

---

A BCPL (*Basic Combined Programming Language*) a B és C nyelvek elődjeként ismert programozási nyelv.

A BCPL sok elemét az Algol60 programozási nyelvtől örökölte.

Jellemző a BCPL nyelvre, hogy a programot általában egy architektúrafüggetlen, INTCODE-nak nevezett közbülső kódra fordították le, amelyet egy gyors értelmező hajtott végre.

# Befunge

A Befunge minden bizonnyal az egyik legbizarrabb programozási nyelv, amelyet valaha megvalósítottak.

A Befunge program egy szimulált automatán fut, amely megvalósít egy vermet és egy kétdimenziós tárterületet.

Gyökvonás (Jason Reed alkotása):

```
v>00p10p>00g:10g\ /v  
  ^ : & <      | : - 1p00 / 2 + <  
> 9 3 * ^      > 0 0 g . @
```

# C

A C programozási nyelv meglehetősen alacsonyszintű nyelv, amely a rendszerprogramozást erőteljesen támogatja. A nyelv elsősorban a UNIX operációs rendszer kapcsán ismert, de sok más rendszer is támogatja.

A nyelv fejlesztőiként Ritchie és Thompson, 1972-73; Kernighan és Ritchie, 1978 ismertek.

A nyelv népszerűségét minden bizonnyal annak köszönheti, hogy a hatékony, hordozható kódot készíthetünk a segítségével. A C hosszú idő óta az egyik legnépszerűbb programozási nyelv.



# C#

A C# programozási nyelvet a Microsoft fejlesztette ki a C, C++, Java és Visual Basic egyes elemeinek ötvözésével.

A fejlesztés elindításának minden bizonnyal van valami köze ahhoz, hogy egy érvényes bírósági döntés (Sun Microsystems vs. Microsoft) értelmében a Microsoft termékeiről el kellett távolítani a Java logót.

Az egyetlen működőképes C# fordítót a Microsoft ingyenesen elérhetővé tette a *.Net Framework SDK* néven, ami sok mindent megmagyaráz.

A C\* a C programozási nyelv párhuzamos számításokra fejlesztett nyelvjárása.

A C\* nyelv alig ismert, dokumentációja nem elérhető és a támogatott párhuzamos számítógéppel együtt valószínűleg kihalásra van ítélve.

# C++

A C++ programozási nyelv a C nyelv objektumorientált programozási módszertant támogató változata.

A C++ nyelvet Bjarne Stoustrup fejlesztette ki az AT&T Bell Labs-nál 1982-85 közt.

A nyelv nagy népszerűségét valószínűleg a fejlett OOP támogatásnak, a C nyelvnek és annak a ténynek köszönheti, hogy a fejlesztés során sikerült megőrizni a C nyelv előnyeit.

A C++ nyelv az egyik legfejlettebb OOP nyelv, amelynek megtanulása igen könnyű a C nyelvet ismerők népes tábora számára.

# Cilk

---

A Cilk a C programozási nyelv párhuzamos programozási elemekkel bővített nyelvjárása, amelyet a M.I.T.-en fejlesztettek (Joerg, Leiserson, et al, MIT, 1995).

A nyelv fordítóprogramja szabadon letölthető és futtatható többprocesszoros UNIX, Linux és Windows NT rendszereken.

# Clean

---

A Concurrent Clean funkcionális programozási nyelv eredeti neve Clean volt.

# CLU

---

A CLU (*Clusters*) programozási nyelv a programozás módszertan „állatorvosi lova”. A nyelvet kísérleti célokra fejlesztették ki, azért, hogy a programozási nyelvek kutatásainak eredményeit kipróbálhassák.

A CLU újdonságainak egy része az őt követő nyelvekbe (Java, Ada) is bekerült.

A SWIFT kísérleti operációs rendszer CLU nyelven íródott.

A PCLU néven ismert CLU fordító ingyenesen letölthető az Internetről.

# COBOL

---

A COBOL (*Common Business-Oriented Language*) az 1960-as évek elejétől igen elterjedt és közkedvelt volt.

A nyelv legfontosabb jellemzői jól használhatóvá tették a banki, üzleti világ számára:

- az angol nyelvhez hasonló nyelvtani szerkezet és kifejezések
- fejlett lista és jelentéskészítő eszközök

A COBOL nyelvet sokszor használták adatbázisokkal összekapcsolva nyilvántartásra, jelentések készítésére.

# COBOL

---

A COBOL nyelvet többször is szabványosították, a mai napig is használják! Közismert tény, hogy napjainkban is több COBOL programsor létezik a világon, mint amennyi az összes egyéb programozási nyelveken megírt sorok száma.

COBOL fordítóprogramokat elsősorban mainframe és UNIX operációs rendszerekre lehet elérni.

Az y2k hisztéria elsősorban a COBOL programokban előszeretettel használt dátumábrázolásra vezethető vissza.



# COBOL

```
IDENTIFICATION DIVISION
PROGRAM-ID. SUM-OF-PRICES.
AUTHOR. TERENCE-PRATT.
SOURCE. PROGRAMMING-LANGUAGES-2ND-EDITION-1984
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INP-DATA ASSIGN TO INPUT.
    SELECT RESULT-FILE ASSIGN TO OUTPUT.
DATA DIVISION.
FILE SECTION.
FD INP-DATA LABEL RECORD IS OMITTED.
01 ITEM-PRICE
    02 ITEM PICTURE X(30).
    02 PRICE PICTURE 9999V99.
    02 FILLER PICTURE X(44).
FD RESULT-FILE LABEL RECORD IS OMITTED.
01 RESULT-LINE PICTURE X(132).
```

# COBOL

```
WORKING-STORAGE SECTION.
```

```
77 TOT PICTURE 999999V99, VALUE 0, USAGE IS COMPUTATIONAL.
```

```
77 COUNT PICTURE 9999, VALUE 0, USAGE IS COMPUTATIONAL.
```

```
01 SUM-LINE.
```

```
    02 FILLER VALUE ' SUM =' PICTURE X(12).
```

```
    02 SUM-OUT PICTURE $$,$$$,$$9.99.
```

```
    02 FILLER VALUE ' NO. OF ITEMS =' PICTURE X(21).
```

```
    02 COUNT-OUT PICTURE ZZZ9.99.
```

```
01 ITEM-LINE.
```

```
    02 ITEM-OUT PICTURE X(30).
```

```
    02 PRICE-OUT PICTURE ZZZ9.99.
```

# COBOL

```
PROCEDURE DIVISION.
```

```
START.
```

```
    OPEN INPUT INP-DATA AND OUTPUT RESULT-FILE.
```

```
READ-DATA.
```

```
    READ INP-DATA AT END GO TO PRINT-LINE.
```

```
    ADD PRICE TO TOT.
```

```
    ADD 1 TO COUNT.
```

```
    MOVE PRICE TO PRICE-OUT.
```

```
    MOVE ITEM TO ITEM-OUT.
```

```
    WRITE RESULT-LINE FROM ITEM-LINE.
```

```
    GO TO READ-DATA.
```

```
PRINT-LINE.
```

```
    MOVE TOT TO SUM-OUT.
```

```
    MOVE COUNT TO COUNT-OUT.
```

```
    WRITE RESULT-LINE FROM SUM-LINE.
```

```
    CLOSE INP-DATA AND RESULT-FILE.
```

```
    STOP RUN.
```

# Common Lisp

---

A Lisp funkcionális (kvázi funkcionális) programozási nyelv, amelynek a Common Lisp a szabványosított nyelvjárása.

A Common Lisp a Lisp programozók 1981-ben indult egységesítési törekvéséből fejlődött ki.

A LISP egyébként egy teljesen őrült, fastruktúrába rendeződő, rengeteg zárójelet használó programozási nyelv, amelyet a mesterséges intelligencia Assembly nyelvének is tekinthetünk.

# Common Lisp

```
;; An example of array search from the common
;; lisp standard
(defun finder (obj vec start end)
  (let ((range (- end start)))
    (if (zerop range)
        (if (eql obj (aref vec start))
            obj
            nil)
        (let ((mid ( + start (round (/ range 2)))))
          (let ((obj2 (aref vec mid)))
            (if (< obj obj2)
                (finder obj vec start (- mid 1))
                (if (> obj obj2)
                    (finder obj vec (+ mid 1) end)
                    obj))))))))
```

# Concurrent Clean

---

A Concurrent Clean tisztán funkcionális, erősen típusos programozási nyelv, amely támogatja a párhuzamos programozást.

A nyelv által támogatott adattípusok: egész, valós, karakterlánc, logikai, listák, tuplek ( $n$ -esek), tömbök és magasabb rendű (matematikai) függvények.

A Concurrent Clean GUI készítését is támogatja.

A nyelv fordítója és részletes dokumentációja az Internetről letölthető.

# Concurrent Clean

```
// Solution to Hamming's problem in Clean,  
// generate an infinite sorted stream of numbers  
// of the form  $(2^n) \cdot (3^m) \cdot (5^p)$ . From the examples  
// distributed with Concurrent Clean v1.2.
```

```
Ham :: [Int]
```

```
Ham = y
```

```
where
```

```
  y = [1:merge (merge (map ((*) 2) y)  
                     (map ((*) 3) y)) (map ((*) 5) y)]
```

```
merge [] [] = []
```

```
merge f [] = f
```

```
merge [] f = f
```

```
merge f=:[a:b] g=:[c:d]
```

```
  | a < c      = [a:merge b g]
```

```
  | a == c     = merge f d
```

```
  | otherwise = [c: merge f d]
```

# Concurrent Clean

---

```
Start :: [Int]
Start = take NrElements Ham

NrElements := 300  // Size of finite sample to take.
```



# Concurrent Pascal

---

A Concurrent Pascal az 1970-es években az operációs rendszerek kutatására készített továbbfejlesztett Pascal változat.

A nyelv új elemei: processzek, monitorok (itt először), párhuzamosítás, osztályok (nincs öröklődés), queue-k.

Az eredeti Pascal elhagyott elemei: rekurzió, goto utasítás, pointerok.

A nyelv komoly hatást gyakorolt az operációs rendszerek és valós idejű rendszerek kutatására.

A C shell a BSD fejlesztési iránya által készített shell, amely elsősorban interaktív parancsfelületként készült, de automatizálásra (programozásra) is használható.

Ugyanakkor a System V ágon inkább a Bourne shell-t favorizálták.

A csh nevét a C programozási nyelvről kapta, mivel a nyelvtana hasonlít a C nyelvtanára.

A csh – mint a legtöbb shell – egyetlen adattípust ismer, a *karakterláncot*, azonban rendelkezik olyan operátorokkal is, amelyek a karakterláncot *számként* kezelik.

A C shell nem tartozik a legelterjedtebb shellek közé, lassan kihalónak is tekinthetjük.

# csch

```
#!/bin/csh
# A simple csh script to find a command on
# the directories listed in the environment
# variable PATH, and print out information
# about it.
set cmd=$1
if (" $cmd" == "") then
    echo "Usage: findcmd commandname"
    exit 1
endif
```

# csch

```
set cnt=0
foreach dir ($path)
    set check="$dir/$cmd"
    if (-x "$check" && ! (-d "$check")) then
        file $check
        ls -ldg $check
        set cnt=$cnt 1
    endif
end

if ($cnt == 0) then
    echo "Sorry, command $cmd not found."
    exit 1
else
    exit 0
endif
```

# cT

---

A cT algoritmikus szkript nyelv, amelyet multimédia prezentációk, animációk, felhasználói felületek készítésére terveztek.

A cT UNIX (X Window) rendszerekre ingyenesen, Windows és Macintosh rendszerekre pénzért érhető el.

# Forth

---

A Forth igen egyszerű, veremszervezési programozási nyelv, amely az 1970-es évek elején mikroszámítógépek és beágyazott rendszerek készítésére terjedt el.

A Forth nyelv egyszerűségéből következik, hogy igen könnyen készíthető kisméretű Forth értelmező. Ez az oka annak, hogy a Forth személyi számítógépek ROM memóriájában is előfordult.

A Forth nyelv egyszerűen bővíthető, a Forth program elolvasása azonban már korántsem ilyen egyszerű. Nem csak azért mert inverz lengyel nyelvtant használ...

# Forth

```
\ Forth implementation of Newton's method for finding
\ roots, simplified. (c) Copyright 1994 Everett F. Carter.
: z1 ( i -- ) ( f: -- z1 )
    z F@ xn{ SWAP } F@ F-
;

: Newton ( i -- ) ( f: e d p -- e d p )
    \ calculate new D
    DUP z1 FROT F* FOVER F+

    \ calculate new P
    FSWAP DUP z1 F* dif{ OVER } F@ F+

    \ calculate new E
    FROT z1 FABS F* FOVER FABS F+

    \ restore stack order
    FROT FROT
;
```

# Forth

```
: FNewt ( &xn &dif n -- ) ( f: z -- e d p)
  >R
  & dif{ &!
  & xn{ &!
  R>

  z F! 0.0e0 0.0e0 0.0e0
  0 DO
    I Newton
  LOOP
;
```



# FORTRAN

---

A FORTRAN minden programozási nyelvek nagyapja, az első programozási nyelv, amely a mai napig is fitt, fejlődik és széles körben használt!

A FORTRAN (*Formula Translation*) nyelvet John Backus vezetésével fejlesztették ki 1954-57 közt az IBM-nél.

A nyelv sok változáson, fejlesztésen esett át, sok programozási nyelvnek (mondhatnánk azt is, hogy minden programozási nyelvnek) az életére hatást gyakorolt. Az egyik legismertebb változata a FORTRAN77 volt.

A FORTRAN nyelvet szabályozó legújabb szabvány a FORTRAN95-öt írja le.

# FORTRAN

---

A FORTRAN fordítók eredetileg lyukkártyákra rögzített programokat hajtottak végre, ezért a sorok hossza nem lehetett nagyobb 72 sornál.

Ez a korlát a legtöbb program és fordító esetében még ma is létezik.

A FORTRAN nyelvben a mai napig is kötelező a címkéket és az utasításokat a megfelelő oszlopokban kezdeni, nem is beszélve a 6. oszlopban található folytatásjelről.

# FORTRAN

---

A régebbi FORTRAN változatok gyakorlatilag csak nagybetűket használtak, azaz a FORTRAN *programban nem szerepelhettek kisbetűk!*

A FORTRAN mai változatai már megengedik kisbetűk használatát, de a FORTRAN fordító ma sem tesz különbséget a kis- és nagybetűk közt.

# FORTRAN

---

A FORTRAN nyelvre nagyon jellemző az implicit típusdeklaráció. Az I, J, K, L, M, N karakterekkel kezdődő nevű változók egészek, a többi pedig valós az alapértelmezés szerint.

Szintén jellemző a tömbök támogatása, a legtöbb FORTRAN programban egészeket, valós számokat és az egészekből és valósokból felépített tömböket használjuk.

A FORTRAN régebbi változataira jellemző volt a GOTO utasítás fejlett használata.

# FORTRAN

```
C MATRIXSZORZAST VEGZO ELJARAS FORTRAN77-BEN.  
  SUBROUTINE MSZORZ (M1,M2,SZ,S)  
    INTEGER S  
    INTEGER M1 (S,S), M2 (S,S), SZ (S,S)  
    INTEGER OSSZ  
    DO 10 I = 1, S  
      DO 10 J = 1, S  
        SZ (I,J) = 0  
        DO 10 K = 1, S  
          SZ (I,J) = SZ (I,J) + M1 (I,K) *M2 (K,J)  
10  CONTINUE  
  END SUBROUTINE
```

# Haskell

---

A Haskell szigorú értelemben véve nem tisztán funkcionális nyelv, amelyet a funkcionális programozással foglalkozó szakemberek egységesítésre való törekvése hozott létre.

A nyelv nevét Haskell B. Curry matematikusról kapta.

A nyelvet elterjedten használják a felsőoktatásban a funkcionális programozási módszertan oktatására.

# Java

---

A Java objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett ki az 1990-es évek közepén.

A Java nyelv nagyon hasonlít a C++ nyelvhez, de átvett elemeket a Modula-3, Mesa és Objective-C nyelvekből is.

A Java standard library package tartalmaz egy security manager nevű komponenst, amely lehetővé teszi, hogy a webböngészők Java alkalmazásokat töltsenek le és futtassanak viszonylag biztonságos módon.

# Java

---

A Java elterjedtségét minden bizonnyal a következő tulajdonságainak köszönheti:

- fejlett és egységes library package
- használható minőségű webböngésző támogatás (beleértve a JavaScript nyelvet)
- a Sun Microsystems eltökélt szándéka és hatalmas anyagi bázisa (ingyen könyvek, ingyen fordító, ingyen SDK, stb.)



# JavaScript

---

A JavaScript programozási nyelvet a Netscape Communications fejlesztette ki az 1990-es évek közepén, eredetileg LiveScript néven.

A nyelv kifejlesztésének célja interaktív webes felületek kialakítása volt, amely megmagyarázza a nyelv hatalmas népszerűségét.

A JavaScript hasonlít a C és a Java nyelvekre, de azoknál egyszerűbb.

A JavaScript tartalmaz ugyan az objektumorientált programozást segítő eszközöket, de nem objektumorientált programozási nyelv.

# JavaScript

---

A JavaScript nyelv fejlődése és élete gyakorlatilag összekapcsolódott a webböngészők fejlesztésével. A nyelv rengeteg olyan eszközt biztosít, amelyek a webes környezetet támogatják.

Ma már gyakorlatilag az összes webböngésző tartalmaz valamilyen JavaScript támogatást.

A Microsoft Internet Explorer nem tartalmaz JavaScript támogatást, JScript támogatást ajánlanak helyette.

# JavaScript

```
<script language=javascript>
sub chkrange(elem,minval,maxval) {
    if (elem.value < minval ||
        elem.value > maxval)
    {
        alert("Value of " + elem.name + " is out of range!");
    }
}
</script>

<input type=button onclick="chkrange(myform.numitem,1,10);">
```

# Lisp

---

A Lisp a mesterséges intelligencia termékeként és eszközeként az 1950-es évek végétől napjainkig igen elterjedten használt.

A Lisp funkcionális programozási nyelv, elterjedten használt a felsőoktatásban.

A Lisp mára igen kipróbált és nagyon jól dokumentált nyelvvé nőtte ki magát, amely rendelkezik azokkal az eszközökkel, amelyekkel a legtöbb imperatív programozási nyelv: objektumorientált eszköztár, GUI eszköztár, adatbázis hozzáférés, és így tovább.

# Logo

---

A Logo a Lisp nyelvvel rokonságban álló funkcionális programozási nyelv, melyet – meglepő módon – gyermekek számítástechnikai oktatásában szokás használni.

A Logo nyelv központi figurája a teknős, amely a képernyőn mozog és rajzol a Logo programban meghatározott módon.

A Logo nyelv feltalálója Seymour Papert, aki a mesterséges intelligencia kutatásának egyik kulcsfigurája.

# Logo

```
; Recursive procedure to line, fractalized
to DrawFractalLine :level :length
  ifelse :level < 1 [
    fd :length] [
    DrawFractalLine (sum -1 :level) (quotient :length 3.00)
    left 60
    DrawFractalLine (sum -1 :level) (quotient :length 3.00)
    right 120
    DrawFractalLine (sum -1 :level) (quotient :length 3.00)
    left 60
    DrawFractalLine (sum -1 :level) (quotient :length 3.00)
  ]
end
```

# Logo

---

```
; procedure to clear screen and position turtle  
to SetupTurtle  
  cs setpos [-160 -10] right 60 clean  
end
```

```
; setup turtle then draw Koch's snowflake(5)  
SetupTurtle  
setpensize [2 2]  
repeat 3 [DrawFractalLine 5 330 right 120]
```

# ML

---

Az ML név mögött a funkcionális nyelvek egész csoportja húzódik meg, mint például ML, SML, SML/NJ, CAML, EML, amelyek közül a legjellemzőbb a Standard ML.

Az ML nevét a Meta-Language kifejezés rövidítéséből kapta.

Az ML tisztán funkcionális nyelvnek tekinthető, amennyiben nem elérhető az értékadás az adattároláshoz.

A legtöbb funkcionális nyelv tudományos vagy elméleti célra készült, nem alkalmazások készítésére, az ML azonban ennek támogatására állomány és hálózatkezeléssel, valamint GUI készítésére alkalmas eszköztárral is rendelkezik.



## Átlag számítása (Andrew Cumming):

```
fun sort nil = nil : int list
|   sort(h::t) = let
    fun insert(i,nil) = [i]
    |   insert(i,h::t) = if i>h then i::h::t else
                        h::insert(i,t)
  in insert(h, sort t) end;

fun mean l = let
    fun sl(nil ,sum,len) = sum div len
    |   sl(h::t,sum,len) = sl(t,sum+h,len+h)
  in sl(l,0,0) end;

mean(sort [2,3,5,7,11,13] @ [6,14,28] )
```

# Modula-3

---

A Modula-3 igen híres, objektumorientált imperatív programozási nyelv, amelyet a DEC fejlesztett ki az 1980-as évek végén.

A Modula-3 a Pascal leszármazottjának tekinthető, a tervezése során a Pascal több vitatható tulajdonságát módosították, a nyelvet bővítették. (Niklaus Wirth is részt vett a fejlesztésben.)

A Modula-3 erősen támogatja a struktúrált programozást, alkalmas nagyméretű szoftverprojektek megvalósítására.

Szakértők szerint igen különös, hogy a Pascal helyett nem a Modula terjedt el...

# Modula-2

---

```
MODULE AlphaRandom;
(* Randomize the alphabet, and show how to use the module Shuffle *)
(* John Andrea, 1992 *)

FROM InOut IMPORT WriteLn, WriteString;
FROM Shuffle IMPORT Deck, Create, Next, Reset;

VAR
    d                :Deck;
    i, j, min, max, n :CARDINAL;
```

# Modula-2

```
BEGIN
  min := ORD( 'a' );
  max := ORD( 'z' );

  n := max - min + 1;
  Create( d, min, max );

  FOR i := 1 TO 10 DO
    WriteString( 'random alphabet = ' );
    FOR j := 1 TO n DO
      WriteString( CHR( Next( d ) ) );
    END;
    WriteLn;
    Reset( d );
  END;

END AlphaRandom.
```

# Objective-C

---

Az Objective-C a C programozási nyelv objektumorientált változata.

Az Objective-C a C++ nyelvvel ellentétben nem tartja meg a C nyelv nyelvtanát, nem támogatja a többszörös öröklődést.

Az Objective-C ugyanazokat a típusokat támogatja, amelyeket a C programozási nyelv.

# Occam

---

Az Occam programozási nyelv a párhuzamos programozás támogatására készült, a transzputerek C programozási nyelvénen (vagy assemblerének) is tekinthető.

A nyelv eredetileg az INMOS Ttransputer chiphez készült az 1980-as évek elején, az „occam for all” projekt keretében azonban bárki számára elérhetővé vált (dokumentációval együtt).

Az occam nyelvben az egyes feladatok „processzek”, amelyek üzeneteket küldhetnek egymásnak és amelyeket sorban vagy párhuzamosan lehet végrehajtani.

# Occam

```
-- Pipelined parallel sort in occam
--(from Pountain and May, A Tutorial
-- Introduction to Occam Programming)
```

```
VAL numbers IS 100 :
[numbers + 1] CHAN OF INT pipe:
PAR
  PAR i = 0 FOR numbers
    input IS pipe[i] :
    output IS pipe[i+1] :
    INT highest :
    SEQ
      input ? highest
      SEQ j = 0 FOR numbers - 1
        INT next:
        SEQ
          input ? next
```

# Occam

```
    IF
      next <= highest
        output ! highest
      next > highest
        SEQ
          output ! highest
          highest := next
SEQ i = 0 FOR numbers    -- get unsorted
  INT unsortednumber :  -- numbers
  SEQ
    input ? unsortednumber
    pipe[0] ! unsortednumber
SEQ i = 0 FOR numbers    -- dump sorted
  INT sortednumber :    -- numbers
  SEQ
    pipe[numbers] ? sortednumber
    output ! sortednumber
```



# Pascal

---

A Pascal programozási nyelvet Niklaus Wirth tervezte az 1960-as évek végén, a 70-es évek elején kimondottan oktatási célokra.

Wirth a struktúrált programozás élharcosa volt, kimondottan furcsa, hogy milyen elnéző volt a nyelv tervezése során.

A legtöbb területen a Pascal után elterjedt nyelvek jobbak, a Pascal mégis mindent túlél az akadémiai szféra támogatásának köszönhetően.

# Perl

---

A Perl nyelvet Larry Wall készítette 1986-ban. A nyelv nevét a *Practical Extraction and Reporting Language* kifejezés rövidítéseként kapta, elsősorban rendszeradminisztrációs és adabányászati célokra készült.

A Perl mára a számítástechnika minden területére betört, igen jellemzők az Internetes alkalmazásai, de mivel gyors alkalmazásfejlesztést tesz lehetővé gyakorlatilag mindent írtak már Perlben.

A Perl szkriptnyelv, bár az újabb változatok a fordítást is támogatják.

# Perl

---

A Perl néhány jellemző előnyös tulajdonsága:

- adattípusok: stringek, számok, listák, asszociatív tömbök
- szabályos kifejezések fejlett támogatása
- objektumorientált eszközök
- kiterjedt system interface
- sokféle vezérlési szerkezet (többféleképpen is lehet...)
- csomagkezelés és rengeteg csomag (tényleg rengeteg)

# Perl

---

Csak egyetlen Perl interpreter létezik, de az szabad és igen jól portolt (UNIX, Linux, Windows, VMS, stb.).

A Comprehensive Perl Archive Network (CPAN) rengeteg Perl bővítményt, programcsomagot tartalmaz, amelyek a számítástechnika szinte minden területét lefedik.

# Perl

```
#!/usr/bin/perl
# Simple program to extract column 3 from a file
# and total up the numbers.
$total = 0;

sub sumcolumn {
    my $col = shift;
    my $lin = shift;
    my @fields;
    if ($lin) {
        @fields = split(/:/, $lin);
        $total = $fields[2];
    }
}

while (<>) {
    sumcolumn(3, $_);
}

print "Total of column 3 is $total\n";
```

# PHP

---

A PHP programozási nyelvet a Webkiszolgálók kiszolgáló oldali programozási nyelveként az 1990-es évek közepén fejlesztették ki.

A nyelv kifejezéseit HTML vagy XHTML nyelvbe lehet ágyazni vagy önálló CGI programként lehet használni.

A PHP hasonlít a shell nyelvekhez, gyengén típusos, kevés típust támogat, egyszerű.

A nyelv hatékony adatbáziseléréssel rendelkezik, ami segíti dinamikus webtartalom előállítását.

## Példa (Yahav Boaz):

```
<?
mysql_connect("localhost","","") or \
    die("Unable to connect to SQL server");
@mysql_select_db("php3") or \
    die("Unable to select database");
$result = mysql_query("select * from customerTable limit 100");
?>
<table border="1">
<tr>
<?
while ($field=mysql_fetch_field($result)) {
    echo "<th>";
    echo "$field->name";
    echo "</th>";
}
```

# PHP

```
echo "</tr>";
while ($row = mysql_fetch_row($result)) {
    echo "<tr>";
    for ($i=0; $i<mysql_num_fields($result); $i++) {
        echo "<td>";
        echo "$row[$i]";
        echo "</td>";
    }
    echo "</tr>\n";
}
echo "</table>";
```



# PL/I

---

A PL/I programozási nyelvet az IBM fejlesztette ki az 1960-as évek közepén. A nyelv nevét a nagyratörő Programming Language 1 rövidítéseként kapta.

Az IBM célja az volt, hogy a mainframe rendszereken minden programot ezen a nyelven írjanak, ami megmagyarázza miért olyan bonyolult a nyelv.

A PL/I a tervezők szándékai szerint egyesíti a FORTRAN, COBOL és Algol jó tulajdonságait.

A rossz nyelvek szerint mára az IBM rengeteg PL/I nyelven megírt programmal rendelkezik és PL/I programozókat keres.

```
FINDSTRINGS: PROCEDURE OPTIONS(MAIN)
  /* READ A STRING, THEN PRINT EVERY */
  /* SUBSEQUENT LINE WITH A MATCH */

  DECLARE PAT VARYING CHARACTER(100),
           LINEBUF VARYING CHARACTER(100),
           (LINENO, NDFILE, IX) FIXED BINARY;

  NDFILE = 0; ON ENDFILE(SYSIN) NDFILE=1;
  GET EDIT(PAT) (A);
```

```

LINENO = 1;
DO WHILE (NDFILE=0);
    GET EDIT(LINEBUF) (A);
    IF LENGTH(LINEBUF) > 0 THEN DO;
        IX = INDEX(LINEBUF, PAT);
        IF IX > 0 THEN DO;
            PUT SKIP EDIT (LINENO,LINEBUF) (F(2),A)
        END;
    END;
    LINENO = LINENO + 1;
END;
END FINDSTRINGS;
```

# Postscript

---

A Postscript programozási nyelvet az Adobe Systems fejlesztette ki nyomtatókkal és más grafikus eszközökkel való kommunikációra az 1980-as évek elején.

A Postscript lapleíró nyelv, de általános programozási nyelvként is használható.

A Postscript veremszervezésű, inverz lengyel jelölést használó nyelv, amelyet viszonylag kevés erőforrással futtathatunk. Hasonlít a Forth nyelvre.

# Postscript

```
%!PS-Adobe-2.0
% Draw a string at an angle at a point
/printat {      % str x y angle => -
    gsave
    4 1 roll translate
    exch rotate 0 0 moveto show
    grestore
} def

% find the center of the page
/pagecenter {   % - => cx cy
    clippath pathbbox 4 1 roll exch sub 2. div
    3 1 roll sub 2. div
} def
```

# Postscript

```
% set up font and constants
/Times-Bold findfont 36 scalefont setfont
pagecenter /cy exch def /cx exch def
/steps 9 def
/basegray 0.75 def
/incrgray basegray steps div def
/baseangle 360 steps div def
/incrangle 360 steps div def
% draw a string as a rosette
steps {
  basegray setgray
  (PostScript) cx cy baseangle printat
  /basegray basegray incrgray sub def
  /baseangle baseangle incrangle add def
} repeat
% done with this page
showpage
```

# Prolog

---

A Prolog (*programming in logic*) deklaratív, logikai programozási nyelvet az 1970-es évek végén, a 80-as évek elején fejlesztették ki.

A Prolog program nem algoritmust ír le, hanem állításokat, tényeket fogalmaz meg, amelyek alapján a Prolog értelmezőnek kérdéseket tehetünk fel. A Prolog értelmező a leírt állításokat alapján univerzális algoritmussal keresi a válaszokat.

Bizonyos problémákra igen gyorsan és egyszerűen készíthetünk Prolog programot, míg más problémák kezelésére a Prolog nem igazán alkalmas.

# Prolog

---

Ha a Lisp nyelvet a mesterséges intelligencia assembly nyelvénnek tekintjük, a Prolog a mesterséges intelligencia BASIC nyelve.

A Prolog nyelv különböző interpreterei mára a nyelv többféle bővítését kezelik, mint például GUI kezelés, adatbáziselérés, képfeldolgozás, objektumorientált eszközök. . .



# Prolog

```
csinos('Gizi').
csinos('Julcsa').
lakik('Gizi', 'Pécs').
lakik('Éva', 'Budapest').

nemszeretkirandulni('Marcsi').
kozelvan('Barcs', 'Pécs').
esik('Budapest').
hideg('Pécs').
esik('Pécs').
otthonmarad(X):- lakik(X,Y), ho(Y).
otthonmarad(X):-nemszeretkirandulni(X).
ho(V):- esik(V), hideg(V).
felhivom(X):-csinos(X),otthonmarad(X),lakik(X, 'Pécs').
felhivom(X):-csinos(X),otthonmarad(X), lakik(X,Y),
               kozelvan(Y, 'Pécs').
```

# Prolog

```
% Quicksort in Prolog, by Keeseey adapted from Bratko
gtq(X,Y) :- X @> Y.

quicksort( [],[] ).
quicksort( [X | Tail], Sorted) :-
    split( X, Tail, Small, Big),
    quicksort( Small, SortedSmall),
    quicksort( Big, SortedBig),
    conc( SortedSmall, [X | SortedBig], Sorted).

split( _, [], [], []).
split( X,[Y | Tail], [Y | Small], Big) :-
    gtq( X, Y),!,
    split( X, Tail, Small, Big).
split( X, [Y | Tail], Small, [Y | Big] ) :-
    split( X, Tail, Small, Big).

conc([],L,L).
conc( [X | L1], L2, [X | L3]) :-
    conc( L1, L2, L3).
```

# Python

---

A Python programozási nyelvet G. van Rossum fejlesztette ki 1991-ben. A nyelv a nevét Monthy Python repülő cirkuszáról kapta, amelyet a szerző munka közben nézett a TV-ben.

A Python magas szintű objektumorientált elemeket is tartalmazó szkriptnyelv széles körű eszköztárral.

A Python eredetileg az Ameoba operációs rendszer számára, mára azonban széles körben használják UNIX, Linux és Windows rendszereken. Windows rendszereken a Python rendelkezik ActiveX támogatással is.

# Python

```
# standard binary search tree from
# a tree data structure package by Dan Stubbs
#
class binary_tree:
    def __init__ (self):
        self.tree = None

    def insert (self, key):
        if self.tree:
            self._insert (self.tree, key)
        else:
            self.tree = node(key)
```

# Python

```
def _insert (self, tree, key):
    if key < tree.key:
        if tree.left:
            self._insert (tree.left, key)
        else:
            tree.left = node(key)
    else:
        if tree.right:
            self._insert (tree.right, key)
        else:
            tree.right = node(key)
```

# QBasic

---

A QBasic más néven QuickBasic a Microsoft által az MS-DOS számára kifejlesztett BASIC nyelvjárása (1986).

A QBasic nagyon sok továbbfejlesztést tartalmaz az eredeti BASIC nyelvhez képest és megdöbbenő módon egészen használható.

A QBasic utolsó változata a 4.5 volt. A Microsoft mára már nem támogatja a QBasic nyelvet, az értelmező/fordító állítólag fellelhető néhány helyen.

# QBasic

---

```
'*  Locates Find$ in sorted array Array$ ()      *  
'*  and returns element number or -1             *  
'*  by Matt Usner.
```

# QBasic

```
FUNCTION BinarySearch% (Array$(), Find$)
    BinarySearch% = -1           ' no matching element yet
    Min = LBOUND(Array$)        ' start at first element
    Max = UBOUND(Array$)        ' consider through last
    DO
        Try = (Max + Min) \ 2    ' start testing in middle
        IF Array$(Try) = Find$ THEN
            BinarySearch% = Try  ' return matching element
            EXIT DO
        END IF
        IF Array$(Try) > Find$ THEN ' too high, cut in half
            Max = Try - 1
        ELSE
            Min = Try + 1        ' too low, cut other way
        END IF
    LOOP WHILE Max >= Min
END FUNCTION
```



# sh

---

A Bourne shell egyike volt a UNIX operációs rendszer első shelljeinek. A nyelvet Steven Bourne fejlesztette ki 1971-ben.

A Bourne shell legfontosabb tulajdonságaiban (típusrendszer, vezérlőszerkezetek, paraméterek, stb.) megegyezik a ma használatos shellekkel.

A legtöbb modern UNIX shell (talán mindegyik) rokonságban áll az eredeti, Steven Bourne által kifejlesztett változattal.

Az sh működését szabványok rögzítik, ezért a legtöbb mai shell képes Bourne shell kompatibilis módban futni.

# sh

---

```
#!/bin/sh
# Simple shell script to count files of size
# greater than 1k

if [ -z "$1" ]
then
    pat="*"
else
    pat="$1"
fi
```

# sh

```
cnt=0
for fname in $pat
do
    if [ -r $fname -a ! -d $fname ]
    then
        size='cat $fname | wc -c'
        if [ $size -gt 1024 ]
        then
            echo $fname '    ' $size
            cnt='expr $cnt    1'
        fi
    fi
done
echo "${pat}: Files bigger than 1K: $cnt"
exit
```

# Simula

---

A Simula programozási nyelvet az 1960-as évek végén fejlesztették ki Norvégiában. A Simula elterjedt változata a Simula67.

A Simula67 volt az első osztályokat is kezelő programozási nyelv, így az objektumorientált programozás úttörőjének tekinthetjük.

A Simula soha nem terjedt el igazán, de a mai napig is változatlan formában létezik és sok programozási nyelvre gyakorolt hatást (pl. C, Java).

A legtöbb operációs rendszerhez ingyenes Simula fordító tölthető le az Internetről.

# Smalltalk

---

A Smalltalk objektumorientált programozási nyelvet a Xerox Software Concepts Group fejlesztette ki 1972-ben.

A Smalltalk híres arról, hogy igen erősen objektumorientált: a nyelvben minden objektum.

A Smalltalk története összeforrott a grafikus felhasználói felület történetével: ez a nyelv volt az első, amely többablakos grafikus felhasználói felületet támogatott.

A fejlesztőcsapat igen komoly hatást gyakorolt a GUI kezdeti éveire.

# Tcl

---

A Tcl igen sikeres programozási nyelv, amelyet a Berkeley-n fejlesztettek ki 1990-ben. A nyelv nevét a *Tool Command Language* kifejezés rövidítéseként kapta.

A Tcl szkriptnyelvet nagyon sokszor az elterjedten használt GUI toolkittel együtt, Tcl/Tk-ként emlegetik.

A Tcl/Tk népszerűségét minden bizonnyal annak köszönheti, hogy a segítségével Windows és UNIX rendszereken egyaránt futtatható GUI-val rendelkező egyszerű alkalmazások készíthetők.

# Tcl

```
# Small Tcl sorting program, after Welch, 1997.

proc NameCompare {a, b} {
    set $asurname = [lindex $a end]
    set $bsurname = [lindex $b end]
    set ret [string compare $asurname $bsurname]
    if { $ret == 0 } {
        $ret = [string compare $a $b]
    }
    return $ret
}
```

# Tcl

```
set namelist {}
set line {}
while { [gets stdin line] != 0 } {
    lappend namelist $line
}
set namelist [lsort -command NameCompare $namelist]
set lineno 1
foreach line $namelist {
    puts stdout "$lineno    $line"
    set lineno [expr $lineno + 1]
}
```



A TeX leíró jellegű nyelv, amelyet dokumentumformázásra fejlesztett ki Donald E. Knuth az 1970-es évek végén.

A TeX nyelvet előszeretettel használják az akadémiai szférában, különösen a matematika területén, mivel a TeX erőssége a matematikai képletek kezelése.

A TeX számára sok makrócsomag készült, amelyek közül az egyik legismertebb a LaTeX (Leslie L. Laport), amely a TeX-el végzett munkát könnyíti meg.

```
% Code to test whether a given year is  
% a leap-year.  From  
% S. Bechtolsheim's "TeX in Practice" volume 3.
```

```
\InputD{imodn.tip}  
\newif\if@LeapYear
```

```

\def\LeapYearConditional #1{ TT\fi
  { \count0 = #1\relax
    \IModN{\count0}{4}{\count1}%
    \ifnum\count1 = 0
      \global\@LeapYeartrue
      \IModN{\count0}{100}{\count2}%
      \IModN{\count0}{400}{\count3}%
      \ifnum\count2 = 0
        \global\@LeapYearfalse
      \fi
      \ifnum\count3 = 0
        \global\@LeapYeartrue
      \fi
    \else
      \global\@LeapYearfalse
    \fi
  }%
  \if@LeapYear
}

```

---

# *Ajánlott irodalom*

---

# Ajánlott irodalom

---

- NYÉKYNÉ GAIZLER JUDIT (szerk.): *Programozási nyelvek*, Kiskapu (2003), ISBN: 963 9301 46 9.
- Az Internet.