



# Programozási nyelvek 1

Szathmáry László  
Debreceni Egyetem  
Informatikai Kar

## 1. előadás

- bevezető
- programozási nyelvek osztályozása
- a Scratch programozási környezet

(utolsó módosítás: 2026. febr. 10.)

2025-2026, 2. félév





# A tantárgyról

**A tantárgy neve:** Programozási nyelvek 1

**A tantárgy kódja:** INBMM0211

**Célcsoport:** MI BSc

**Az előadó honlapja:** <https://arato.inf.unideb.hu/szathmary.laszlo>

(Google: „Szathmáry DEIK”)

**Az előadás ideje és helye:**

- kedd 8.00, IK-201

# Követelmények

## Gyakorlat

A gyakorlati aláírás megszerzésének egyik feltétele a **rendszeres részvétel** a gyakorlatokon. A félév során legfeljebb 3 hiányzás megengedett. Aki ezt túllépi, az nem kap aláírást. A hiányzásokat pótolják be!

A gyakorlatra mindenki hozzon **jegyzetfüzetet**!



A szorgalmi időszakban **2 zárthelyi dolgozat** lesz, mindenki számítógép mellett. A ZH-kat legalább 50%-osra kell teljesíteni (külön-külön).

Egy ZH-n kb. 4 feladat várható; ezek közül legalább kettőt tökéletesen kell megoldani. Egy sikertelen ZH-t egyszer meg lehet majd ismételni.

Minden gyakorlat végén kapnak majd házi feladatokat. Ha valaki a **házi feladatok** 75%-ánál kevesebbet old meg, az nem kap aláírást.

# Követelmények

## Előadás

Vizsga: a félév végén online teszt lesz.

Az előadásokon **erősen ajánlott** a részvétel!

# Ajánlott irodalom

- Brian W. Kernighan, Dennis M. Ritchie: A C programozási nyelv. Műszaki Könyvkiadó, 2. kiadás, 2008 !
- Ivor Horton: Beginning C. Apress, 5. kiadás, 2013
- Peter van der Linden: Expert C Programming: Deep C Secrets. Prentice Hall, 1. kiadás, 1994
- C Notes for Professionals, <https://goalkicker.com/CBook/>
- Nyékyné Gaizler Judit (szerk.): Programozási nyelvek. Kiskapu Kft., Budapest, 2003
- Juhász István: Magas szintű programozási nyelvek 1. mobiDIÁK könyvtár, egyetemi jegyzet, 3. kiadás, 2008 !
- Kósa Márk, Pánovics János: Példatár a Programozás 1 tárgyhoz. mobiDIÁK könyvtár, egyetemi jegyzet, 1. kiadás, 2004
- Juhász István, Kósa Márk, Pánovics János: C példatár. Panem Könyvkiadó, 2004

# Bevezető

- Miről szól a programozás?



- Miről szól ez a tárgy?
- Mi a kapcsolat az „Adatszerkezetek és algoritmusok” c. tárggyal?

Niklaus Wirth:

„Algoritmusok + adatszerkezetek = programok”

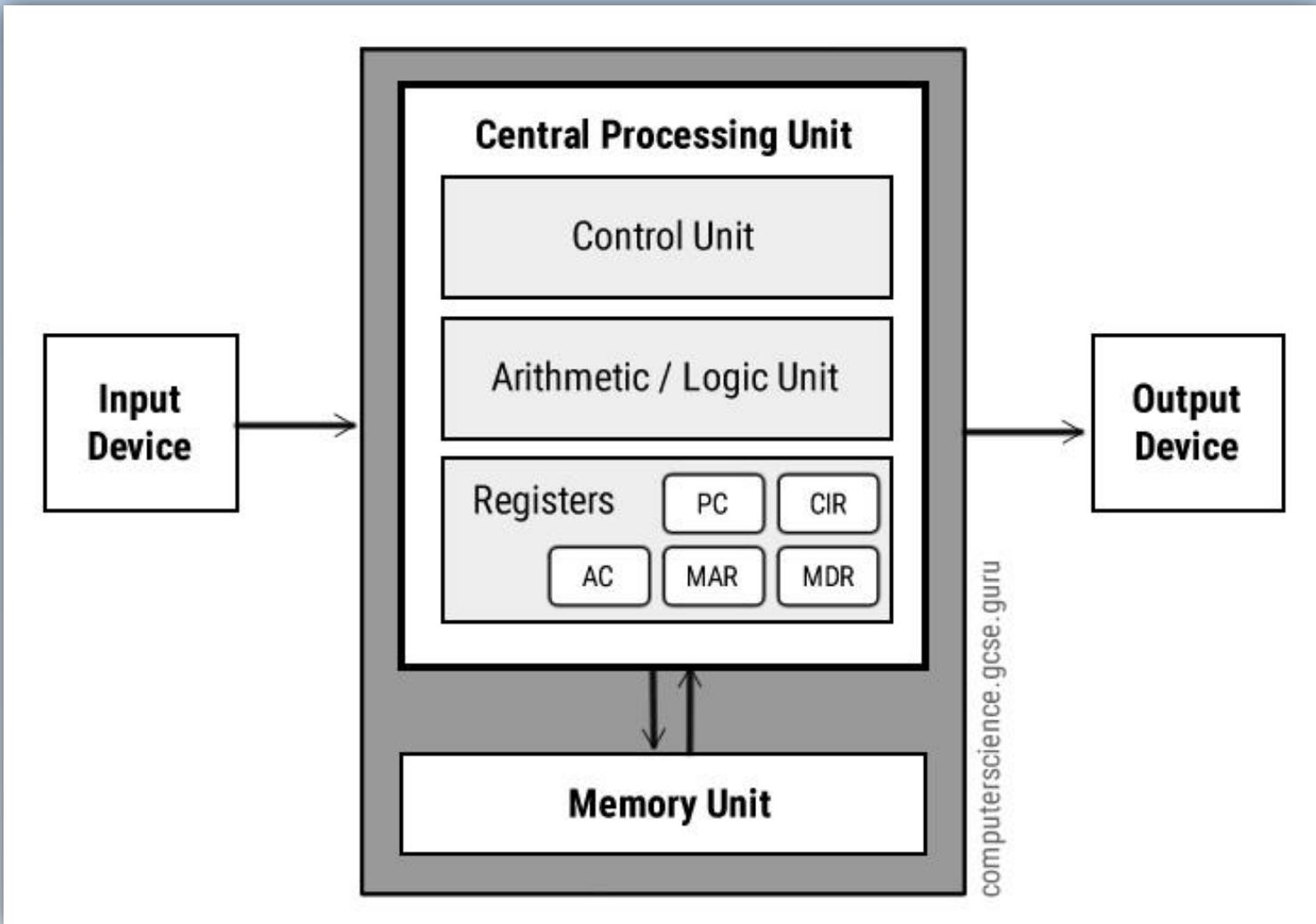
# Számítógép-architektúra

A programozási nyelvek többségét az uralkodó számítógép-architektúra köré tervezik, amit manapság Neumann-féle architektúraként ismerünk.

Az imperatív nyelvek a legdominánsabbak a Neumann-elvű számítógépek miatt:

- az adat és a program a memóriában tárolódik
- kettes számrendszer használata
- a memória elkülönül a CPU-tól
- az utasításokat és az adatokat el kell juttatni a memóriából a CPU-hoz (busz)
- az imperatív nyelvek alapjául szolgálnak
  - a változók modellezik a memóriacellákat
  - az értékadó utasítások modellezik az adatszállítást

# A Neumann-architektúra



# Számrendszer, számábrázolás

- 10-es számrendszer, pl. 125
- inkrementálás, túlcsordulás
- 2-es számrendszer; bit, byte
  - 00011011 , mennyi ennek az értéke?
  - 1 byte-on (8 biten) 256 különböző érték tárolható ( $2^8$ )
    - előjel nélkül: 0 – 255
    - előjelesen: -128 – 127



# A programozási nyelvek evolúciója

# Gépi kód

mem.-cím	gépi kód	assembly utasítások
0044CA49	50	push eax
0044CA4A	42	inc edx
0044CA4B	00A850420020	add [eax+\$20004250], ch
0044CA51	56	push esi
0044CA52	42	inc edx
0044CA53	00A85542002C	add [eax+\$2c004255], ch
0044CA59	60	pushad
0044CA5A	42	inc edx
0044CA5B	00E8	add al, ch
0044CA5D	5F	pop edi
0044CA5E	42	inc edx
0044CA5F	00CC	add ah, cl
0044CA61	7243	jb +\$43
0044CA63	00647243	add [edx+esi*2+\$43], ah
0044CA67	00C4	add ah, al
0044CA69	854300	test [ebx+\$00], eax

# A programozási nyelvek evolúciója

## Gépi kód

- A program **számok** sorozata
- 1 utasítás = 1 szám
- 1 memória cím = 1 szám
- Nincsenek változók
- Nincsenek ciklusok
- Nincsenek eljárások



- rendkívül gyors
- minimális memóriahasználat
- adott platformhoz (CPU-hoz és hardver elemekhez) optimalizálható



- rendkívül nehéz írni, olvasni, módosítani
- a program CPU-függő

# A programozási nyelvek evolúciója

## Assembly nyelvek

mem.-cím	gépi kód
0044CA49	50
0044CA4A	42
0044CA4B	00A850420020
0044CA51	56
0044CA52	42
0044CA53	00A85542002C
0044CA59	60
0044CA5A	42
0044CA5B	00E8
0044CA5D	5F
0044CA5E	42
0044CA5F	00CC
0044CA61	7243
0044CA63	00647243
0044CA67	00C4
0044CA69	854300

## assembly utasítások

```
push eax
inc edx
add [eax+$20004250], ch
push esi
inc edx
add [eax+$2c004255], ch
pushad
inc edx
add al, ch
pop edi
inc edx
add ah, cl
jb +$43
add [edx+esi*2+$43], ah
add ah, al
test [ebx+$00], eax
```

# A programozási nyelvek evolúciója

## Assembly nyelvek

- **Mnemonikok** kódolják az utasításokat (rövid, könnyen megjegyezhető szócskák)
  - pl. MOV = „move”, INC = „increase”
- 1 utasítás = 1 mnemonik
- Könnyebb írni / olvasni / módosítani a forráskódot
- A program CPU-függő

# A programozási nyelvek evolúciója

## Assembly nyelvek

Új fogalmak jelennek meg:

- **Forráskód:** a program szöveges specifikációja
- **Fordító:** alkalmazás, mely lefordítja a forráskódot
- **Gépi kód:** a forráskód fordításának eredményeképp áll elő; a CPU közvetlenül tudja futtatni



# A programozási nyelvek evolúciója

## Assembly nyelvek

Megjelennek:

- **Primitív típusok:** csak memóriaigény, nincs szemantika
- **Primitív változók:** nincs típusellenőrzés, hatáskör és élettartam menedzsment
- **Primitív ciklusok:** feltételes ugrás a kód egy korábbi pontjára
- **Primitív eljárások:** hívás, visszatérés, nincs standard paraméterátadás

# A programozási nyelvek evolúciója

## Procedurális (eljárásorientált) nyelvek

- A cél, hogy általános célú eljárásokat / függvényeket írunk
- Standard mód a paraméterezésre
- Standard mód az értékkel való visszatérésre
- Példák: Basic, Pascal, C

# A programozási nyelvek evolúciója

## Procedurális (eljárásorientált) nyelvek

**3 alapvető programozási szerkezet:**

- szekvencia
- szelekció
- ciklus

**Utasításblokkokat** is ki tudunk alakítani a változók élettartam- és hatáskör-menedzsmentjéhez.

# A programozási nyelvek evolúciója

## Procedurális (eljárásorientált) nyelvek

Megjelennek az „igazi” típusok:

- Egyszerű típusok, mint pl. bool, char, int, float, stb.
- Összetett típusok, mint pl. tömbök, rekordok, stb.
- A nyelvek többségénél:
  - minden egyes változóhoz típust kell rendelnünk (ezt változódeklarációnak nevezzük)
  - változók értékkadásakor típusellenőrzés

# A programozási nyelvek evolúciója

## Procedurális (eljárásorientált) nyelvek

Megjelennek az „igazi” **eljárások/függvények**:

- **Formális paramétereket** lehet megadni
- Automatikusan ellenőrzi a rendszer, hogy az **aktuális paraméterek** illeszkednek-e a formális paraméterekre

# A programozási nyelvek evolúciója

## Procedurális (eljárásorientált) nyelvek

- Könnyebben írható / olvasható
- Gyorsabban írható és módosítható, hiszen általános célú eljárásokat használunk. Az eljárások könnyen újrahasznosíthatók.
- Biztonságosabb
- Könnye(bbe)n lehet platformfüggetlen kódot írni. A forráskódot természetesen újra kell fordítani egy új platformon.

# A programozási nyelvek evolúciója

## Objektum-orientált (OOP) programozási nyelvek

- Adatok és eljárások/függvények szorosan integrálódnak
- Saját típusok definiálhatóak, korlátok nélkül
- Könnyebb a való világot modellezni
- Példák: SmallTalk, C++, Java, C#
- Lásd Programozási nyelvek 2. (köv. félév)

# A programozási nyelvek evolúciója

## Specjalizált nyelvek

- Speciális célra való nyelvek
  - pl. adatbázis-kezelés (pl. SQL), matematika (pl. R), grafika, ipari robotok programozása
- Könnyű olvasni és megtanulni
- (Elméletileg) még nem hozzáértők is felfogják

# A programozási nyelvek evolúciója

## Mesterséges intelligencia (MI) nyelvek

- Majdnem mint a természetes nyelvek
- Egy tudásbázist kell megadni
- Aztán lekérdezéseket írni
  
- Példa: Prolog

# Programozási paradigmák

**Imperatív nyelvek:** A forráskód utasítások sorozata, melyet a számítógép végrehajt. Leírjuk, hogy pontosan **hogyan** akarjuk megoldani a feladatot. Algoritmusokat írunk. Példák: procedurális és OOP nyelvek.

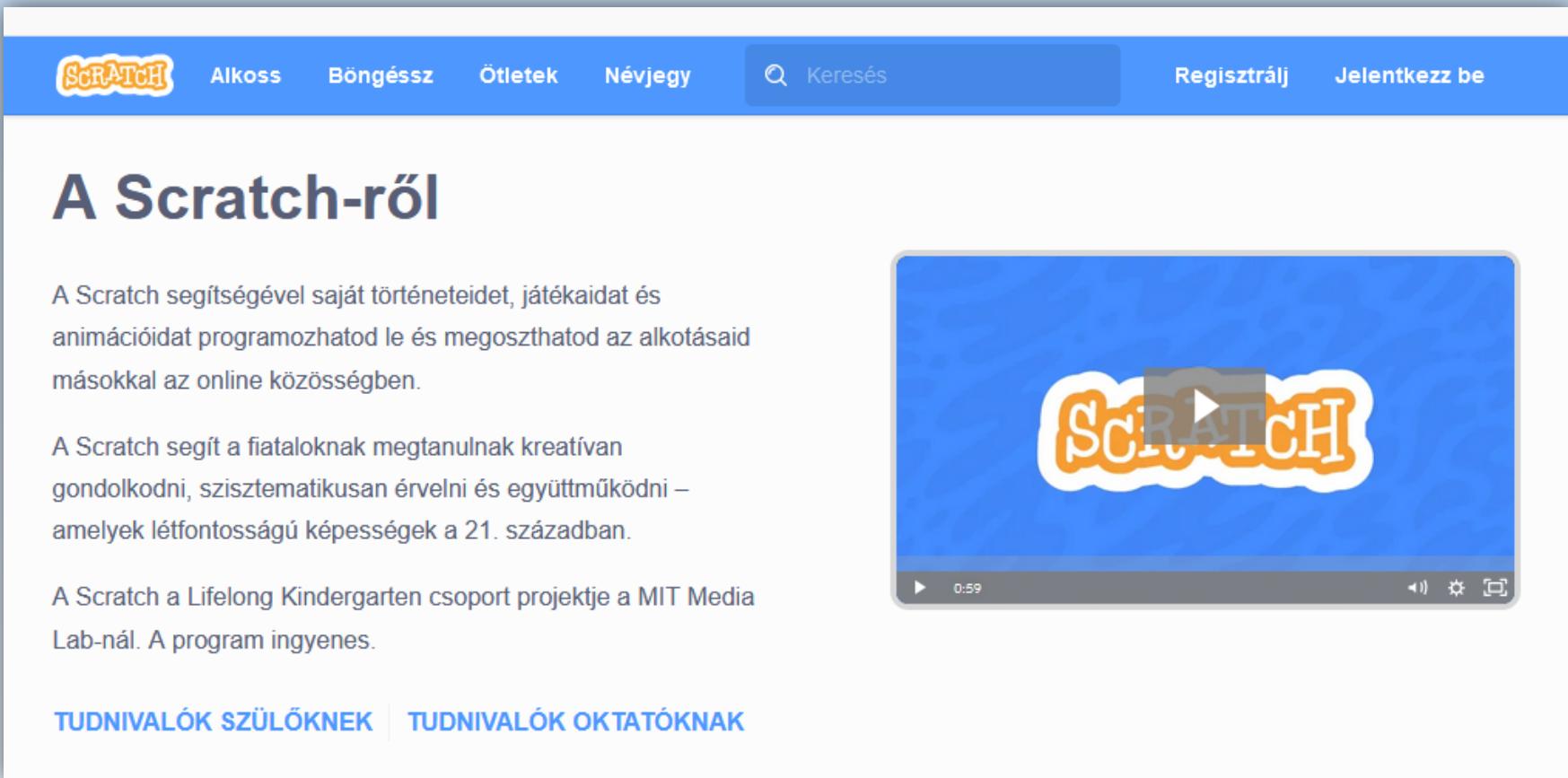
**Deklaratív nyelvek:** A forráskód azt specifikálja, hogy **mit** akarunk megoldani. (És nem azt, hogy **hogyan**.)

- **Funkcionális programozás:** A program egy hatalmas kiértékelendő függvény. Példa: LISP
- **Logikai programozás:** A program logikai kifejezésekből áll. Példa: Prolog

**Multi-paradigmás nyelvek:** A fenti paradigmák keverednek. Példák: Python, R, Rust, C#, Java, stb.

# Scratch -- demó

<https://scratch.mit.edu>



The screenshot shows the official Scratch website (<https://scratch.mit.edu>). At the top, there's a navigation bar with links for 'Alkoss', 'Böngéssz', 'Ötletek', 'Névjegy', a search bar ('Keresés'), and user options ('Regisztrálj', 'Jelentkezz be'). Below the navigation, the main title 'A Scratch-ről' is displayed in large, bold, dark blue text. To the right of the title is a video player window showing the Scratch logo on a blue background. The video player has a play button, a timestamp of '0:59', and standard media control icons.

**A Scratch-ről**

A Scratch segítségével saját történeteidet, játékaidat és animációidat programozhatod le és megoszthatod az alkotásaid másokkal az online közösségen.

A Scratch segít a fiataloknak megtanulnak kreatívan gondolkodni, szisztematikusan érvelni és együttműködni – amelyek létfonosságú képességek a 21. században.

A Scratch a Lifelong Kindergarten csoport projektje a MIT Media Lab-nál. A program ingyenes.

**TUDNIVALÓK SZÜLŐKNEK** | **TUDNIVALÓK OKTATÓKNAK**

# Házi feladat

- Szerezzék be a K & R-féle „C Bibliát” és kezdjék el tanulmányozni (bevezetés, 1. fejezet). **Mindent próbáljanak ki!** Ez nem egy regény!
- Kezdjenek el ismerkedni a **Linux** operációs rendszerrel. Tekintsék meg ezeket a videókat: <http://bit.ly/36UAL8D> , és **próbáljanak ki mindennt**.
- Regisztráljanak a **GitHub** oldalon (<https://github.com>).
- Látogassák meg és fedezzék fel a [https://github.com/jabbalaci/Programozas\\_1](https://github.com/jabbalaci/Programozas_1) repository-t.
- Készítsünk egy saját Scratch projektet! ([feladat leírása](#))

# Szorgalmi

- Olvassuk el: *Már kisgyerekként hatjegyű számokat osztott el fejben Neumann János* ( <https://bit.ly/31G7PjO> )
- Olvassuk el: *Aki gyorsabban számolt, mint a saját számítógépe – Neumann János, az informatikai forradalom elindítója* (<https://bit.ly/3Id5zWX> )