

## **A programnyelvek osztályozása**

### **Imperatív nyelvek:**

– Algoritmikus nyelvek: a programozó mikor egy programszöveget leír, algoritmust kódol, és ez az algoritmus működteti a processzort. – A program utasítások sorozata. – Legfőbb programozói eszköz a változó, amely a tár közvetlen elérését biztosítja, lehetőséget ad az abban lévő értékek közvetlen manipulálására. Az algoritmus a változók értékeit alakítja, tehát a program a hatását a tár egyes területein lévő értékeken fejti ki. – Szorosan kötődnek a Neumann-architektúrához.

Alcsoportjai:

– eljárásorientált p. ny.: a forráskód szövege részekre tagolható, melyeket eljárásoknak nevezünk; általában fordító programosak, csak ritkán interpreteresek. Pl. C, C++, Pascal, Fortran

– objektum-orientált p. ny.: kielégíti az OOP alapelveket (lásd lejjebb). Pl. C++, Java, Perl

– objektum-alapú p. ny.: az öröklés és a sokalakúság elvén kívül minden OOP alapelvet kielégít.

Pl. JavaScript

### **Deklaratív nyelvek:**

– Nem algoritmikus nyelvek.

– Nem kötődnek olyan szorosan a Neumann-architektúrához, mint az imperatív nyelvek. – A programozó csak a problémát adja meg, a nyelvi implementációkba van beépítve a megoldás megkeresésének módja. – A programozónak nincs lehetősége memóriaműveletekre, vagy csak korlátozott módon.

Alcsoportjai:

– funkcionális (applikatív) p. ny.: a program végrehajtását egy függvény kiértékeléseként tekintik: a legtisztább funkcionális nyelvekben nincsenek változók, nincs értékadás, a függvényeknek nincsegyáltalán semmi mellékhatásuk. A függvények nem tesznek mást, mint a bemeneti paraméterek egy halmaza alapján kiszámítanak egy outputot. A funkcionális paradigma azt diktálja, hogy a megoldandó problémát funkcionális egységekre kell bontanunk – olyan függvényekre, amelyek egy megadott bemenethez egy kimenetet rendelnek. a forráskód hossza a procedurális nyelvekben írt

megoldásnál lényegesen rövidebb lehet, ugyanakkor, a hibaarány is radikálisan csökkenthető.

Ellenben, bizonyos problémák, amelyek procedurális nyelvekben kézenfekvően megoldhatók, funkcionális nyelvekkel nehezen kezelhetők. pl. LISP

– logikai p. ny.: a matematikai logikára épül; pl. PROLOG, LISP

### **Máselvű nyelvek:**

Olyan nyelveket sorolunk ebbe a kategóriába, amelyek máshová nem sorolhatók.

Nincs egységes jellemzőjük. Általában tagadják valamelyik imperatív jellemzőt. pl. FORTH, LOGO

### **További osztályozási szempontok lehetnek:**

- típusos nyelvek: létezik a változó fogalma, melynek lényeges attribútuma a típus, mely meghatározza, hogy a változó milyen értékeket vehet fel. Az eljárásorientált nyelvek általában típusosak
- nem típusos nyelvek: ha létezik is a változó fogalma, a típus komponens nem létezik. általában a deklaratív nyelvek.
- leíró (markup) nyelvek: ún. „markup” szimbolikát használ, vagyis <TAG>adat</TAG> formában adhatóak meg az utasítások. Pl. HTML, XML, SQL, TEX
- szkriptnyelvek: ún. szkripteket használ, melyek futásidőben kerülnek értelmezésre, azaz előzetes gépi kódra fordítás nélkül beolvassa őket az értelmező környezet, majd a lefuttatás idején utasításról utasításra végrehajtatja azt az operációs rendszerrel. pl. awk, lua, JavaScript, PHP, Python

**Fordítás:** A fordító tehát az a program, ami egy adott nyelvű forráskódot egy gépi kódra lefordít, vagyis előállítja azt az utasítássorozatot a célnyelven, ami ugyanazt a programot eredményezi. A lefordított forráskódot a számítógép egy adattárolóba (nem csak a memóriába) elmenti, ami ezután hordozható, vagyis átmásolva más gépekre (amiknek ugyanaz az utasításkészlete) végrehajtható, futtatható. A fordítást csak egyszer kell végrehajtani, de minden egyes platformra.

**Interpretálás:** egy másik módszer nem gépi nyelven írt forráskód futtatására. Ennek lényege, hogy amikor a forráskódot szeretnénk végrehajtani, akkor annak utasításait egyenként beolvassuk, az adott utasítást lefordítjuk gépi kódra és egyből végrehajtjuk. Ezután folytatjuk a következő utasítással. A lefordított kód ebben az esetben csak a memóriában tárolódik, tehát az egyes utasítások fordítását, vagyis az egész folyamatot minden egyes végrehajtásnál újra el kell végezni. Az interpretáláshoz szükség van egy programra, ami a fenti műveleteket elvégzi, ezt hívjuk interpreternek. Az interpreternek minden egyes gépen léteznie kell, amin a programunkat futtatni szeretnénk, viszont nem kell a fordítással nekünk bajlódni, hanem a platformfüggetlen forráskódot tudjuk hordozni. Az interpretálással futtatott programok még lassabbak, hiszen a fordítást is a futtatás közben végezzük el.

**Implementáció:** Egy algoritmus, architektúra, szabvány, modell, specifikáció vagy egyéb terv konkrét megvalósítása. Az implementáció a számítástechnikában egy technikai specifikáció vagy algoritmus program, program komponens vagy más módon történő megvalósulása.