Alapok

*Ha már programoztál valaha, a Programok és Programozási nyelvek részt nyugodtan kihagyhatod. Ezekben a részekben azokhoz szólok, akik **tényleg** sosem programoztak semmit.*

Programok

Anélkül, hogy tudj számítógépet kezelni, aligha tudnád ezt a szöveget elolvasni. Bizonyára járatos vagy a Word, Excel, webböngésző programok (Firefox, Chrome, Edge…) valamelyikének a használatában, tudsz programokat telepíteni, és mappákat, fájlokat kezelni.

Biztosan arról is hallottál, hogy a programok „érthetetlen jelekből" (gépi kód – machine code) állnak, és hogy a számítógép ezek érthetetlen jeleket értelmezi, és hajtja végre. Meg hogy a számítógépek nullákkal és egyesekkel dolgoznak.

A számítógép alapvetően csak nagyon egyszerű műveletekre képes, mint összeadni két számot, beolvasni egy adatot a memóriából, kiírni egy adatot a memóriába, stb.

A programozók közül sokan zsenik, de azért nem annyira, hogy ezeket az érthetetlen jeleket írogassák egymás mögé. Az ő munkájukat további programok segítik. A programozók programfejlesztő környezeteket (integrated development environment, IDE) használnak általában. Ide bepötyögik ember számára értelmezhető módon azt, amit a számítógéppel meg akarnak csinálni, aztán ebből a program elkészíti az „érthetetlen jeleket", a gépi kódot.

A programok készítését nevezzük *programozás*nak (programming).

Programozási nyelvek

Az ember számára értelmezhető leírása a programnak valamilyen *programozási nyelv*en (programming language) történik. Az idők során több száz programozási nyelv alakult ki, melyek bizonyos problémára hatékonyabb, más jellegű problémákra kevésbé hatékony megoldást adtak. Az egyik pl. matematikai számítások programozására volt kiváló, a másikban üzleti alkalmazásokat vagy éppen adatbázissal dolgozó programot lehetett jól írni. Meg voltak *általános célú* programozási nyelvek (general purpose programming language), amikben mindent egyformán nehéz volt *leprogramozni* (vagyis megírni). A Java egy *általános célú* programozási nyelv.

A programozási nyelvek bizonyos szempontból hasonlítanak az emberi nyelvekre. Pl. ahogy aki magyarul tud, nem érti meg automatikusan az angol nyelvet, de pl. aki spanyolul tud, elboldogul az olasszal is, mert azok hasonlítanak egymásra. Így a programozási nyelvek között is vannak olyanok, amelyeket egy másiknak a tudásával meg lehet érteni. Persze ahogyan egy spanyol sem feltétlen ért meg egy olasz nyelvű tudományos vagy filozófiai értekezést, úgy egy programozási nyelven megírt összetettebb megoldásokat sem fogja megérteni az, aki másik programozási nyelvet ismer. A legtöbb ma széles körben használt programozási nyelv olyan, mintha egy

nyelvcsalád tagjai lennének – ha ismered az egyiket, nem nulláról kell kezdened a másikat. És amúgy a programozási nyelveknek is ugyanúgy van nyelvtana (*szintaxis* - *syntax*), ahogy az emberi nyelveknek.

Ráadásul, ahogy haladt előre az informatika világa, a korábbi tapasztalatokat fel- és a számítógépek egyre nagyobb kapacitását kihasználva egyre fejlődtek a programozási nyelvek is. Ma már divat az, hogy inkább a gép dolgozzon sokat, az ember meg minél kevesebbet. Az ember munkabére drágább, mint nagyobb, gyorsabb számítógépet venni, így gazdasági, üzleti szempontból ez a döntés racionális. Egyre *magasabb szintű, absztraktabb* programozási nyelveket alkottak meg. Ez azt jelenti, hogy azok a fogalmak, amiket a programozási nyelv használ, sokkal bonyolultabbak, nagyon messze állnak attól, mint amit a számítógép használ, így magának a fejlesztőkörnyezetnek sok-sok munkával le kell egyszerűsítenie olyan szintre, amit a számítógép is megért. A Java egy *magas szintű* programozási nyelv.

Programozási nyelvre néhány példa: C, C++, C# (ejtsd: szí-sárp), PHP, Javascript, Java, Python, Visual Basic.

Vannak olyan programozási nyelvek, amiket egy program átír (lefordít - compile) gépi kódra, amit aztán az adott gép végre tud hajtani (pl. C, C++), vannak olyanok, amiknél egy program „szinkrontolmácsolja”, értelmezi (interpret) a megírt programot a számítógépnek (PHP, Javascript, Python). Az analógiából is láthatod, hogy a lefordítás által kapott eredmény gyorsabb és jobb, mint a szinkrontolmácsolás. Aztán vannak a hibrid megoldások, mint amit a Java is nyújt: részben lefordítja (egy nem létező, azaz *virtuális* számítógépre), amit aztán szinkrontolmácsolja a konkrét számítógépre, amin fut. Ennek következtében oldható meg az, hogy a lefordított program többféle számítógépen is tudjon lefordított állapotában futni. Ez amúgy nem ennyire kézenfekvő. Ahogyan nem is olyan régen minden mobiltelefonhoz saját töltőcsatlakozó volt, úgy gépi kód is minden számítógéphez saját volt, az egyik nem volt *kompatibilis* a másik rendszerrel.

Sokszor persze megoldható volt az, hogy az eredeti programodat, az ún. *forráskód*ot (amit te írsz; *source code*) átrakod egy másik gépre, majd azon a gépen is elkészíted a gépi kódot… De a szupertuti új alkalmazásunkat csak úgy odaadjuk egy teljesen ismeretlennek? Nem-nem! Az üzleti titok. Valami megoldás kell ahhoz, hogy egy másik *platform*on is futhasson… De mi is az a platform?)

Platformok

Az elmúlt évtizedek során nemcsak programozási nyelvekből, hanem számítógéptípusokból is többféle jött létre. Olyan, mint manapság az iPhone és az Androidos telefonok: más gyártó gyártja őket, másképp működnek, egymás programjait nem tudják futtatni, de mégis mindkét környezet (ún. *platform*) ugyanarra való: hogy pörgesse az ember a Facebookot a metrón. ☺

Bonyolítja a helyzetet, hogy az asztali számítógépek és laptopok világában még egy tényező bejön, az ún. *operációs rendszerek* (operating system) fogalma. Feltételezhetően Windows-t használsz – vagy ha nem, úgyis tudod, hogy mi az. ☺ Ez az operációs rendszer, egy alapprogram, amely a többi programnak nyújt különféle szolgáltatásokat: kezeli a winchestert/SSD-t, memóriát, különféle külső eszközökhöz biztosít hozzáférést (egér, nyomtató,…), stb. stb. Ha az operációs rendszer nem azonos két gépen, akkor szintén külön *platform*nak tekinthetjük, általában nem tudják egymás programjait futtatni. Más operációs rendszerek a Windows mellett, amiről lehet, hogy már hallottál, de ha nem, mindenképpen érdemes: Linux, MacOS. Ezeken kívül még több tíz vagy száz operációs rendszer van, de leendő junior programozóként elég, ha ezekről tudsz valamit, illetve a Windows és Linux kezelésében nem árt járatosabbnak lenni, mert legtöbb munkahelyen ezek egyike vagy másika az, amit használnak.

Mit lehet programozni és miben?

Az egyik legkézenfekvőbb terület, hogy a programokat asztali számítógépen vagy laptopon (notebook, ultrabook,…) futtatod. Ezt összefoglalóan csak asztali számítógépeknek (desktop computer) fogom nevezni. Mi először ezt fogjuk megtanulni, méghozzá Java nyelven, mert ez a legegyszerűbb rész. Asztali számítógépekre való (*desktop*) programok írására a Java mellett jó még a C, C++, C#, Python, Visual Basic…

Az okostelefonokra és tabletekre írt szoftverek egy másik nagy csoport. Androidra jelenleg Javában (de gyorsan terjed a Kotlin), iOS-re (Apple) pedig Objective-C-ben és Swiftben, Windows Phone-ra pedig C++-ban, C#-ban, Visual Basic-ben… lehet programokat írni.

A harmadik a webes szoftverek fejlesztése. Ez alatt bármilyen olyan (ún. dinamikus) honlapot kell érteni, ahol nem csak az oldalakat lehet letölteni, hanem valamit „csinálni” is: ilyenek a webáruházak, internetes banki felületek, mindenféle megrendelős felületek. Ez az oldal, a StudiCore Online is egy ilyen. Webes környezetben jól használható nyelvek: Java, PHP, Javascript, C#.

Ezen kívül vannak speciális ún. *beágyazott rendszerek (embedded systems)*, amelyek valamilyen készülék vezérléséért felelősek, legyen az öntözőberendezés, mosógép, TV, mikrohullámú sütő. Ezeket a készülékeket leginkább C, C++ nyelveken programozzák, mert limitált számítási és tárolási kapacitásuk van, nem éri meg minden készülékbe 10-szerannyi memóriát és 10-szer gyorsabb feldolgozóegységet tenni… Tudod, amikor a gombok számán is spórol a gyártó, ott nem várható el ilyesmi. A C és C++ segítségével (bár több munkával, de) lehet gyorsabb, kevesebb memóriát használó programot írni.

Itt említem meg a Microsoft egy széles körben használt programozási környezetét, a .NET-et (ejtsd: dot-net). A *programozási környezet*et (computing platform) úgy képzeld el, mint egy szerszámosláda, amiben rengeteg, előre megalkotott eszközt találsz a különböző feladatokra. Csak a programozásban nem fúró, kalapács és csavarhúzó van a szerszámosládában, hanem ablakok kirakására alkalmas utasítások, hálózatkezeléshez kapcsolódó műveletek, stb. Az is fontos, hogy a szerszámosládák, akarom mondani programozási környezetek egymással nem

felcserélhetők (*nem kompatibilisek*), ami olyan, mintha a Bosch szerszámkészletével kifúrt lyukba (bocsánat, furatba) csak a Bosch kalapácsával lehetne tiplit beütni, a Siemens-évl nem. (Vagy a Samsung robotgéppel elkészített tésztát csak Samsung sütőben lehetne kisütni).

A .NET programozási környezettel tudsz programot írni desktop környezetre, de tudsz webes programot is és Windows telefonos applikációt is készíteni.

Ezen bevezető után lássuk, mi is a Java. Megfogalmaztam egy szép mondatot, amitől nem kell megijedni, minden részét megmagyarázom:

Java nyelv

A Java egy C alapokon nyugvó, objektum-orientált, platformfüggetlen programozási nyelv és környezet.

Mit is jelentenek ezek?

- *Programozási nyelv*: Erről már beszéltünk.
- *C alapokon nyugvó*: A C programozási nyelvet 1970-ben adták ki, majd újabb és újabb nyelvek születtek belőle (C++, PHP, C# és maga a Java). Az alap nyelvi elemek nagyon hasonlóak az eredeti C-ben használtakra, csak bővültek, egyszerűsödtek. Így azt lehet mondani, hogy aki tud C-ben/C++-ban/PHP-ban/C#-ban programozni, az a Java alap utasításait (változók, elágazások, ciklusok, tömbök – tanuljuk majd) elég gyorsan át tudja majd futni. (Megjegyzés. Közkeletű hiedelem, hogy mivel a C volt az alap, ezért azt célszerű először megtanulni, majd utána a C++-t, majd a C#-ot és a Javát. Kis túlzással ez olyan, mint ha valaki előbb lovagolni tanulna meg, hiszen először az volt a helyváltoztatás leggyorsabb eszköze, majd gőzmozdonyt vezetni, majd utána ül csak autóba).
- *Objektum-orientált (object oriented)*: Ez egyfajta programozási stílus, ami mostanában nemcsak cool dolog, hanem megkerülni sem igazán lehet. A Java azért jó, mert kellően rigorózus nyelv ahhoz, hogy „kikényszerítse" a precíz gondolkodást, az elvek helyes alkalmazását ebben a stílusban, ilyen szempontból egy nagyon jó tanulónyelv – kezdetben ez inkább segítség, mint hátrány. Az, hogy konkrétan mi az, hogy objektum-orientált programozás, hogy kell úgy programozni, azt a 12. fejezettől kezdjük majd el tanulni.
- *Platformfüggetlen (platform independent)*, azaz az elkészített programot nem csak a saját számítógépeden, mondjuk Windows alatt tudod futtatni, hanem a kész programot odaadva a linuxos vagy Mac-es barátodnak, ő is fogja tudni futtatni. Persze ehhez kell egy extra szoftvert telepítenie (a neve JRE), de attól még tök jól működik.
- *Programozási környezet*: a Java nyelvhez is készítettek egy szabványos „szerszámosládát" (könyvtár – library), amiből kedvünkre válogathatunk. Néha olyan szinten, hogy a programfejlesztés nem is a program megírásából, hanem a megfelelő szerszámok megtalálásából és összeillesztéséből áll. (Illetve ha még tovább megyünk, és idevesszük az összes elérhető szerszámosládát a Javához, akkor egy idő után a szerszámosládák keresgélésével és megismerésével megy el a

programfejlesztésből egy jelentős időszelet – és még így is megéri, mert nem kell újra elkészíteni magunknak az adott eszközöket. Ebben a tanfolyamban a végén megnézünk két ilyen szerszámosládát, hogy megtapasztald, hogy hogyan is kell egy szerszámosládát megismerni).