ELSŐ FEJEZET

Mi a programozás?

Ha volt már a kezünkben programozásról szóló könyv, akkor pontos képünk lehet arról, hogy mi a programozás. Végül is, miért akarnánk valamit tanulni, ha fogalmunk sincsen, hogy mi is az valójában? Mégis, a programozás világának újoncai, vagy a programozás művészetét gyakorló szakemberek, akik egy adott szakma révén már rendelkeznek egy kevés programozási ismerettel is, hasznosnak találhatják a következő rövid összefoglalást a programozás történetéről, arról, hogy tulajdonképpen mi a programozás, és hol tart ma.

A programozás története

A programozás története jóval nagyobb múltra tekint vissza, mint azt sokan gondolnánk. Rengetegen úgy vélik, hogy a programozás a huszadik század végének találmánya. Valójában a modern programozás és a programozási nyelvek története az 1940-es évek közepére, 60 évre nyúlik vissza.

Mielőtt azonban elkezdenénk a történet bemutatását az 1940-es évektől, menjünk vissza az időben egy kicsit messzebbre. Egészen 1822-ig és Charles Babbage-ig. Az angliai Cambridge egyetem hallgatójaként Babbage felismerte, hogy az akkori számítóeszközök nagy része, például a csillagászati táblázatok, vízállásdiagramok, hajózási térképek kritikus hibákat és hiányosságokat tartalmaznak. A hibák miatt sok hajó eltévedt a tengeren, és ez gyakran a legénység életébe és a rakományba került. Mivel a pontatlanságok forrásának az emberi tényezőt tartotta, az volt az elképzelése, hogy gőzgépek segítségével a táblázatok és a diagramok készítéséből és karbantartásából kiiktatja a találgatásokat. Az egyik ilyen gép, a Difference Engine néven emlegetett szerkezet Babbage életének hátralevő részében rengeteg idejét lekötötte. Még a brit kormányt is felkereste, hogy pénzügyi segítséget kérjen: az első (de véletlenül sem az utolsó) kéréssel a számítógép-tudományi kutatás finanszírozásához kért kormányzati támogatást.

A Difference Engine 10 évnyi fejlesztése után Babbage felismerte, hogy az egyfunkciós gép kizárólag egyetlen művelet végrehajtására képes. Rájött, hogy ez igen komoly korlátozás, és egy időre felfüggesztette a munkát a sokoldalú Analytical Engine fejlesztésének kedvéért. Ez a gép a modern számítógép alapelemeit tartalmazta, és megalapozta Babbage számára a „számítógép atyja” nevet.

Az Analytical Engine nem nyert széles körben elismerést, mivel Babbage ugyanazzal a problémával küzdött, ami évszázadokra megkeserítette a programozók és az informatikusok életét: képtelen volt világosan dokumentálni és ezáltal közkinccsé tenni ötleteit!

Az Analytical Engine fejlesztése 1842-ig folytatódott, amikor is a brit kormány belefáradt abba, hogy nem lát előrelépést és eredményeket, és megvonta Babbage-től az anyagi támogatást. Babbage ennek ellenére egészen 1847-ig folytatta a munkát, és ekkor visszatért a Difference Engine fejlesztéséhez. 1847 és 1849 között 21 részletes rajzot készített a szerkezet második verziójának megépítéséhez. Mindamellett Babbage egyik gépet sem fejezte be. 1871-ben bekövetkezett halála után fia, Henry Prevost a Difference Engine egyszerű aritmetikai egységének több másolatát is elkészítette, és elküldte különböző intézményeknek szerte a világban – egyet a Harvard egyetemre is eljuttatott –, hogy a gép az utókor számára fennmaradjon.

A fejlődés az 1800-as években folytatódott, és 1854-ben Charles Boole elkészítette a nevét viselő szimbolikus logikai rendszert (Boole-logika), amelyet mind a mai napig használunk (és a könyvben is részletesen foglalkozunk majd vele). A rendszer bevezette a „nagyobb, mint", „kisebb, mint", „egyenlő" és „nem egyenlő" fogalmát, és egy szimbolikus rendszer segítségével megjelenítette ezeket a fogalmakat.

A mondás szerint a szükség találékonnyá tesz, és a szükség 1890-ben állt elő, amikor az Egyesült Államok kongresszusa több kérdést szeretett volna hozzáadni a népszámlálási nyomtatványokhoz. Az Egyesült Államok lakosságának növekvő száma azt jelentette, hogy az adatok feldolgozása egyre tovább tartott, és becslések szerint az 1890-es népszámlálási adatok feldolgozása nem fejeződött volna be az 1900-as népszámlálás előtt, hacsak valamilyen módon nem lehetett volna felgyorsítani a folyamatot.

A népszámlálási hivatal versenykiírásának az volt a célja, hogy fellendítse a számítástudomány iránti érdeklődést, és ennek eredményeként előálljon egy adatfeldolgozó berendezés, amely a kormány munkáját segíti. A versenyt Herman Hollerith nyerte, és miután bebizonyította a technológia eredményességét, más országokban is vállalta népszámlálási információk feldolgozását, majd később megalapította a Hollerith Tabulating Co. vállalatot. (Ez a vállalat egyike volt annak a három cégnek, amelyek egyesülésével 1914-ben létrejött a CTR, a Calculating Tabulating Recording Company. A név nem feltétlenül ismerős, de 10 évvel később a vállalat International Business Machines, azaz IBM néven működött tovább, ezt a nevet már egészen biztosan hallottuk!)

> *Hollerith nem csak ezen a téren volt az első: az ő gépei jelentek meg először az újságok címlapján.*

A fejlődés ekkor kissé lassulni látszott, és az 1920-as évek közepéig a számítástechnikai eszközök használata az üzleti életben, az iparban és a mérnöki tudományokban még nem terjed el. Ami azt illeti, a legáltalánosabban használt eszköz az egyszerű analóg logarléc volt. A dolgok 1925-ben gyorsultak fel ismét, amikor a Massachusetts Institute of Technology (MIT) alkalmazottja, Vannevar Bush kifejlesztette differenciálanalizátorát, amely integrálási és differenciálási képességekkel rendelkezett. A fejlesztést a Rockefeller Alapítvány támogatta, és 1930-ban ez volt a világ legnagyobb „számítógépe".

A programozás történetének következő nagy alakja a német Konrad Zuse. 1935-ben Zuse kifejlesztette Z-1 névre hallgató számítógépét. Annál a ténynél, hogy a számítógépet szülei nappalijában építette meg, csak az volt említésre méltóbb, hogy ez volt az első számítógép, amely reléket használt, és a kettes számrendszer segítségével számolt. A maga nemében az első ilyen számítógép volt a számítógépek modern korának előfutára.

Zuse folytatta munkáját, és Helmut Schreyer segítségével 1938-ban kifejlesztette Z-2 számítógépét. A német kormány anyagi támogatását kérte a gép fejlesztéséhez és megépítéséhez, de kérését a kormány elutasította, mert a projekt tovább tartott volna, mint amennyi ideig – a tervek szerint – a háború. A háború végén Zuse először Hintersteinbe, majd Svájcba menekült, ahol a zürichi egyetemen újraépítette Z-4 gépét.

Zuse volt az, aki a világ első programozási nyelve, a Plankalkül kifejlesztésével 1946-ban megalapozta a modern programozást. A Z-3 számára olyan kódot írt, amely lehetővé tette, hogy sakkozzon a géppel. A nyelv megjelenése azért volt nagy áttörés, mert modern nyelvek összetevőinek nagy részét, többek között a táblákat és az adatstruktúrákat is tartalmazta.

Zuse később saját vállalatot alapított, amely végül a Siemens Corporation része lett.

Az 1945-ös év a számítástechnika történetében újabb fontos állomást jelentett, felbukkant az a szó, amelytől mindenki hidegrázást kap: a hiba! 1945-ben Grace Murray Hopper (később Hopper admirális) a Harvard egyetemen a Mark II Aiken-féle átkapcsoló kalkulátoron dolgozott. A gépekkel akkortájt folyamatosan probléma volt, és az egyik ilyen kellemetlen eset során, 1945. szeptember 9-én az egyik technikus felfedezte, hogy egy moly került a gép áramköreibe (a korabeli feljegyzés szerint a szakember az F panel 70-es reléjében találta a lepkét). A technikus eltávolította a molyt, és beragasztotta abba a naplóba, amelyben a számítógép használatával és problémáival kapcsolatos megjegyzéseket rögzítette. Az esettel kapcsolatban a következő naplóbejegyzés született: „Az első eset, amely során tényleges hibát (bug) találtam." Ebből a bejegyzésből született a „gép hibakeresése" (debugged), valamint a „számítógép hibakeresése" (debugging a computer) és a „számítógépprogram hibakeresése" (debugging a computer program) kifejezés is.

> *Bár Grace Hopper mindig elismerte, hogy nem volt ott, amikor az eset történt, ez volt az egyik kedvenc története, amelyet gyakran felemlegetett.*

Ekkor a dolgok kezdtek igazán felgyorsulni. 1949-ben megjelent a *Short Code* programozási nyelv. A kódot kézzel kellett géppel olvasható kóddá változtatni (a *fordítási* folyamat segítségével), ezért igazából kevés dolog volt a nyelvvel kapcsolatban igazán „rövid".

1954-ben az IBM elkezdte a FORTRAN (a *FORmula TRANslator* rövidítése) programozási nyelv fejlesztését. A FORTRAN 1959-ben került piacra, és a könnyen használható bemenet és kimenet rendszerének, illetve a rövid és követhető kódnak köszönhetően azonnal hatalmas sikert aratott (és sok helyen még ma is használják). A FORTRAN volt az első magas szintű kereskedelmi programozási nyelv, ami azt jelenti, hogy a kód könnyebben olvasható, mert ismerős szintaxisszabályokat, nyelvtant és struktúrát alkalmaz.

1958-ban az FORTRAN II és az ALGOL is megjelent, megkezdődött a LISP fejlesztése, 1959-ben pedig egy újabb népszerű és hosszú életű nyelv, a COBOL (Common Business Oriented Language) kelt életre a Data Systems and Languages konferencián (CODASYL). A COBOL kereskedelmi intézményi nyelv, amelyet elsősorban nagyszámítógépeken használnak, és ma is sok vállalat alkalmazza.

A fejlődés nyaktörő sebességgel folytatódott, rengeteg új nyelv jelent meg, és az már létező nyelvek újabb és újabb változatai és verziói követték egymást. 1968-ban megkezdődött a *Pascal* fejlesztése is. A Pascal 1970-ben jelent meg, és oktatási céllal ma is széles körben használják. 1970-ben két olyan nyelv is elérhetővé vált, amelyek forradalmasították a számítástechnikát. Ez a két nyelv a Smalltalk és a B-nyelv volt. A Smalltalk megjelenése azért volt fontos, mert a nyelv teljes mértékben objektumokon alapul (még ne aggódjunk, ha nem tudjuk mi az objektum), a B-nyelv pedig megjelenésének következményei szempontjából jelentett mérföldkövet.

Milyen következményei voltak a B-nyelvnek? 1972-ben Dennis Ritchie létrehozott egy B-nyelven alapuló nyelvet, amelynek a C nevet adta (miután a nyelv egy darabig NB névre hallgatott). A C az egyszerűséget, a hatékonyságot és a rugalmasságot hozta magával, és a programozás új korszakának volt a hírnöke, amelyben a korábbiakhoz képest hatékonyabban, gyorsabban és egyszerűbben lehetett dolgozni.

1975-ben megjelent dr. Wong Tiny BASIC programozási nyelve. A TinyBASIC használatához csupán 2 KB memóriára volt szükség, és papírszalag segítségével kellett a számítógépbe tölteni. Azért hozott áttörést, mert ez volt az első freeware (használata ingyenes volt, nem kellett fizetni érte) program.

> *Az „All Wrongs Reserved" és „Copyleft" szabadalmi sorokkal ebben a programban találkozhattunk először.*

Az 1975-ös évben írta meg és adta el a MIT-nek az akkor ifjú Bill Gates és Paul Allen a BASIC nyelv saját verzióját.

A nyaktörő sebesség az 1970-es, 1980-as és 1990-es években is folytatódott. Egyre több és több fejlesztés látott napvilágot és vezetett a ma tapasztalható sokszínűséghez: rengeteg különböző programozási nyelvhez, amelyek mindegyike rendelkezik egyaránt előnyös és hátrányos tulajdonságokkal. Láttuk az internet megjelenését, amely szintén nyelvek seregét hozta magával. Az internet másik nagy előnye, hogy megkönnyíti az információk és a programok megosztását másokkal, tehát a programozás és a programozási nyelvek iránti érdeklődés egyre nő, és ezt az érdeklődést az információk, ötletek és alkalmazások akadálytalan cseréje még inkább fellendíti.

Rövid utazást tettünk a programozás történetében, kiemelve a főbb eseményeket, amelyek a mai helyzet kialakulásának mérföldkövei voltak. Ideje, hogy megtudjuk, valójában mi is a programozás.

Mi a programozás?

Erre a kérdésre nem létezik egyszerű válasz. A programozás mindenki számára mást és mást jelent. Azonban sikerült megfogalmaznom a következő meghatározást, amely nagyjából fedi az igazságot:

A programozás az a képesség, amely révén a számára érthető nyelven beszélhetünk a számítógéphez olyan nyelvtan és szintaxis segítségével, amelyet követve a számítógép számunkra hasznos feladatokat hajthat végre.

Egyszerűen összefoglalva, erről szól a programozás. Megírjuk a kódot, a számítógép értelmezi a kérést, és csinál valamit.

A „csinál valamit" a programozás létfontosságú része. Mindig azért programozzuk a számítógépet, hogy csináljon valamit (még akkor is, ha a tevékenység az újabb utasításra való várakozást jelenti!). A programozás célja, hogy csináljunk valamit, és haladjunk előre. Amikor programozunk, nem számít, hogy a feladat egyszerű vagy bonyolult, mindig valamilyen utasítást adunk a számítógépnek. Rendszerint egyszerre csak egy utasítást. Bár úgy tűnik, hogy a számítógéptől „egyidejűleg" több, különböző feladat végrehajtását kérjük, valójában lépésenkénti utasításokkal látjuk el a teendőket illetően.

A kódnak helyesnek és egyértelműnek kell lennie. Nem tartalmazhat hibákat vagy félreérthető utasításokat. Ezek bármelyike miatt a kód kudarcot vallhat, és hibaüzenetek jelenhetnek meg. A számítógép nem rendelkezik megfelelő eszközökkel ahhoz, hogy kitalálja, mi a kód célja, illetve menet közben kijavítsa az esetleges hibákat a kódban.

A programozás elsajátításának kezdetén ismerjük fel, hogy ritkán írunk olyan kódot, amely csak és kizárólag egyetlen dolgot csinál. Még a legegyszerűbb projektek is több lépésből állnak:

1. A program elindítása.
2. A kezdeti paraméterek ellenőrzése.
3. A paraméterek módosítása.
4. A futtatás utáni takarítás.
5. A program kilép.

Miért van szükség ennyi programozási nyelvre?

Ha a számítógépnek olyan formában adunk utasításokat, amelyet megért, miért van mégis ennyi különböző programozási nyelv? Biztosan olyan kódot írunk, amelyet a számítógép megért?

Igaz, hogy a különféle számítógépek különböző típusú kódot értenek meg (a PC alapjaiban különbözik a Macintoshtól), de az egyes számítógéptípusok valójában egy nyelvet értenek meg, és ez nem az a nyelv, amely segítségével a kódot írjuk. A kód, amelyet beírunk, nem az a kód, amelyet a számítógép megért. Egy program (egy parancsértelmező vagy fordító) szükséges ahhoz, hogy a kódot átalakítsa bináris nyelvre, amelyet a számítógép megfejt. Ez a számítógépes nyelv legnagyszerűbb része.

A számítógép bináris utasítások olvasása révén működik. A bináris rendszer már ismerős lehet számunkra. A tízes alapú rendszertől különböző számrendszer, amelyben csak két számjegy van: 0 és 1.

Decimális számjegyek: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Bináris bitek: 0, 1

Decimális	Bináris
0	0
1	1
2	11
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

1.1. ábra

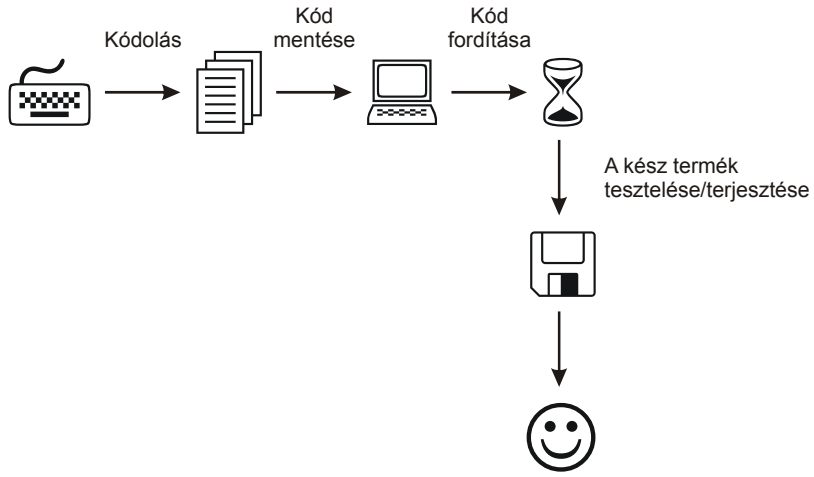
A bináris kódolás fárasztó, és ha a számítógéppel mindig bináris üzemmódban kellene dolgoznunk, a dolgok tényleg bonyolulttá válnának (bár a billentyűzet nagyban leegyszerűsödne).

Vizsgáljuk meg közelebbről az 1.2. ábrát!

01000010011010011011011100110000101110010011110010010000001101001011100110010000
00111010001100101011001000110100101101111011101010111000110010000001100001011011100
01100100001000000110100101100110001000000111011101100101001000000110100001100
00001101001011011100010000001100010011010010110111001100001011100100111100100100
00000111011101101001011101000110100000100000110111101110101011100100010000011
00011011011101101101011100000111010101110100011001010111001001110011001000000111
01010111001101101001011011100110011100100000110100101110100001000000110000101 1
01000110110000100000011101000110100001100101001000000111010000110100101101101 01
10010010000000111010001101000011010010110111001100111011100110010000001110111 01
10111101110101011011000110010000100000011001110110010101110100001000000111011001
10001010111001001111001001000000110001101101111011011010111000001101100011010010 11
00011011000010111010001100101011001000010000001101001011011100110010001100101 01
10010101100100001000000010100001100001011011000111010001101000011011110111010 10
11001110110100000100000011011110111010101110100001000000110101101100101011110010 1
10001001101111011000010111001001100100001110011001000000111011101101111011101010 11
01100011001000010000001100010011001010010000001100001001000000110110001101111
01110100001000000111001101101001001101101011100000110110001100101011100100001000 0
1001010010010110

1.2. ábra

Mit jelent az 1.2. ábra? Ez az utolsó bekezdés, amelyben az ASCII- (American Standard Code for Information Interchange) rendszer betűit bináris számrendszer segítségével jelenítettük meg. Nem kell aggódnunk, hogy még nem értjük, ez mit jelent (a későbbiekben visszatérhetünk, és újra elolvashatjuk a fenti sorokat), most elég annyi, ha hálásak vagyunk azért, hogy a kódolás nem nullák és egyesek végtelen sorainak begépelését jelenti.



1.3. ábra

A programozás, illetve a kódolás során tulajdonképpen olyan kódot írunk, amelyet egy másik program megért, és olyan kódra tudja fordítani, amit pedig a számítógép ért meg; innen az *értelmező* fogalma. Tehát, amikor a kódot írjuk, nem a számítógép szabályait kell követnünk, sokkal inkább az értelmező (vagy hasonlóképpen a fordító) szabályait, amelyek a számítógép számára is érthető formára alakítják a kódot.

Az 1.3. ábrán a kódolási folyamat egyszerűsített diagramját látjuk.

Különböző kód, azonos eredmény

Annak bemutatására, hogy különböző kóddal ugyanazt az eredményt érhetjük el, nézzük végig a következő kódot különféle programozási nyelveken. Néhányukról talán már hallottunk is.

BASIC

```
10 print "Helló, világ!"
20 goto 10
```

Atari BASIC

```
10 REM HELLO.BAS
20 POKE 764,255
30 PRINT "Helló, világ!"
40 IF PEEK(764)=255 THEN GOTO 30
```

C

```
#include <stdio.h>

main()
{
for(;;)
{
printf ("Helló, világ!\n");
}}
```

C++

A C++-kód régebbi verziója:

```
#include <iostream.h>

main()
{
for(;;)
{
cout << "Helló, világ! ";
}}
```

A kód új verziója:

```
#include <iostream>

int main()
{
std::cout << "Helló, világ!\n";
}
```

COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300 DATE-WRITTEN. 02/09/04 17:24.
000400* SZERZŐ FRED F
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500 DISPLAY "HELLÓ, VILÁG!" LINE 15 POSITION 10.
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.
```

FORTRAN

```
C
c Helló, világ!
C
Program Hello

implicit none
logical DONE

DO while (.NOT. DONE)
write(*,10)
END DO
10 format('Helló, világ!')
END
```

Java

```
class HelloWorld {
public static void main (String args[]) {
for (;;) {
System.out.print("Helló, világ!");
}
}
}
```

JavaScript

```
<title>
Hello World in JavaScript
</title>
<script>
alert("Helló, világ!")
</script>
```

Mathematica

```
While[True, Print["Helló, világ!"]]
```

Pascal

```
Program Hello (Input, Output);

Begin
Writeln ('Helló világ!');
End.
```

Perl

```
print "Helló, világ!\n" while (1);
```

Python

```
while (1) :
print "Helló, világ!";
```

QBASIC

```
begin:
print "Helló, világ!"
goto begin
```

Smalltalk

```
Transcript show:'Helló, világ!';cr
```

Visual Basic

```
Private Sub Form_Load()
Static I
I = 1
for I = 1 to 10
msgbox "Helló, világ!"
Next I
end sub
```

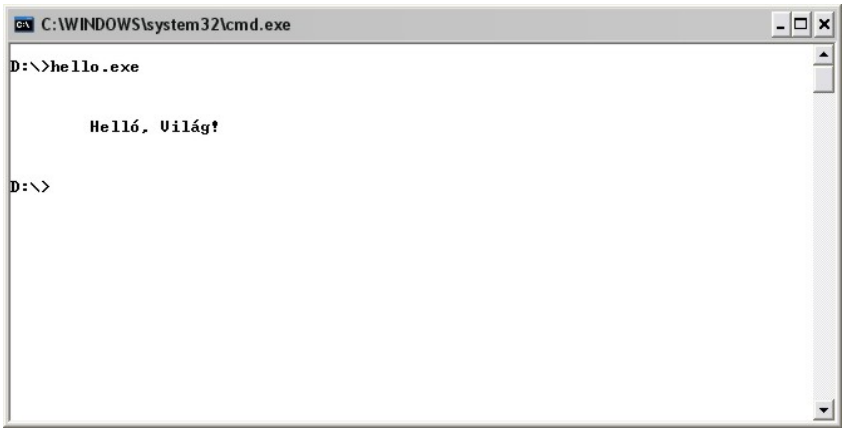
VRML

```
#VRML V1.0 ascii

AsciiText {
string "Helló, világ!"
justification LEFT
}
```

A különféle nyelveken megírt kódok egyetlen feladatot hajtanak végre: a képernyőn a „Helló, világ!” feliratot jelenítik meg.

Az 1.4. ábrán a C++-kód eredményét látjuk:



1.4. ábra

Az 1.5. ábrán a JavaScript-kód feliratát tekinthetjük meg:

1.5. ábra

Miért éppen „Helló, világ!”?

Miért a „Helló, világ!”-ot használjuk a programozási példákban? A példa eredetét és történetét sűrűn átszövi a városi legenda és mítosz. Úgy tűnik, hogy az első „Helló, világ!”-ot alkalmazó gyakorlat a C-nyelvet megelőző B-nyelv esetén készült.

Ezt megelőzően ez a példa szolgált annak jelzésére, hogy minden beállítás rendben van, és a nyelv működik. Tehát, miért is szakítanánk a hagyománnyal?

A programok létrehozásához szükséges programok

A semmiből létrehozhatunk programokat? Ha muszáj, létrehozhatunk, hiszen a programozás úttörői is így cselekedtek, de az első programozási nyelvek megjelenése után a számítástechnikusok első dolga az volt, hogy olyan programokat készítsenek, amelyek megkönnyítik a további programok írását. Miután ez megtörtént, az adott nyelven sokkal könnyebben és gyorsabban lehet kódolni. Hamarosan mindenki egy nyelvet alkalmazott a teljesen különböző (vagy új) nyelvekkel együttműködő eszközök fejlesztése során. Ez lehetővé tette, hogy az új nyelvek összetettebbek és kifinomultabbak legyenek, és ezáltal még gazdagabb funkcionalitást biztosítsanak.

A programok két típusa segíti a kódolást és programkészítést:

- a fejlesztői környezetek,
- a fordítók.

A korábban említett értelmező nem a kód megírásához, csak annak futtatásához szükséges.

Fejlesztői környezet

A fejlesztői környezet az a program, ahová a kódot begépeljük. Néhány nyelv speciális fejlesztői környezetet követel meg (például a Visual Basic), míg más nyelvek esetén csupán egy egyszerű szövegszerkesztő szükséges (a JavaScript például olyan nyelv, amelynek használatához semmi más nem kell, mint egy szövegszerkesztő). Több nyelvre is igaz, hogy esetükben válogathatunk a használni kívánt fejlesztői környezetek között (a C++ is ilyen nyelv).

A fejlesztői környezetekről a későbbiekben részletesen szót ejtünk, most elég annyit tudnunk, hogy léteznek ilyen környezetek.

Fordítók

A fordító olyan program, amely a begépelt kódot a számítógép számára érthető kódra módosítja. Ez a folyamat a *fordítás*.

A fordító elolvassa a begépelt kódot, ellenőrzi az esetleges hibákat, és meggyőződik arról, hogy a kód megfelel a szabályoknak, és van értelme. Ha a kóddal kapcsolatban valamilyen hibát talál, a fordítónak (ha jó fordítóról van szó) erről értesítenie kell bennünket, és le kell állítania a fordítás folyamatát. Ha nem talál semmilyen problémát, a fordító előállít egy önmagában futtatható (vagy végrehajtható) fájlt, és végrehajtja a kódnak megfelelő utasításokat. A személyi számítógépeken ezt a fájlt általában futtatható fájlnak nevezik, és a fájl *.exe* kiterjesztéssel rendelkezik.

1.6. ábra

Összegzés

A fejezetben megtudtuk, hogy valójában mit is jelent a programozás. A fejezet elolvasása után még nem tudunk programozni, de a következő területeket már (ha röviden is) megtárgyaltuk:

- A programozás története.
- A programozási nyelvek alapjai.
- Gyors bepillantás egy kevéske kódba.
- A programozás folyamatának áttekintése a kódtól a futtatható fájlig.
- Néhány fontosabb programozási fogalom.

A következő fejezetben megtudhatjuk, miért van *igazán* szükségünk arra, hogy megtanuljunk programozni!