PYTHON

A PROGRAMKÉSZÍTÉS FOLYAMATA

1. **Specifikáció**: A *megrendelő* kitalálja, hogy milyen programot szeretne. A *programozó* beszélgetni kezd a megrendelővel. Megpróbálják pontosítani, mit vár el a megrendelő, mit kell tudnia a programnak. A tisztázott igények precíz leírását nevezzük *specifikációnak*.

2. **Tervezés**: A megoldandó feladathoz megfelelő *adatszerkezeteket* és *algoritmusokat* kell találni, vagy fejleszteni. Megtervezendő továbbá a program felhasználói felülete, és gondolni kell a jövőbeli bővíthetőségre is.

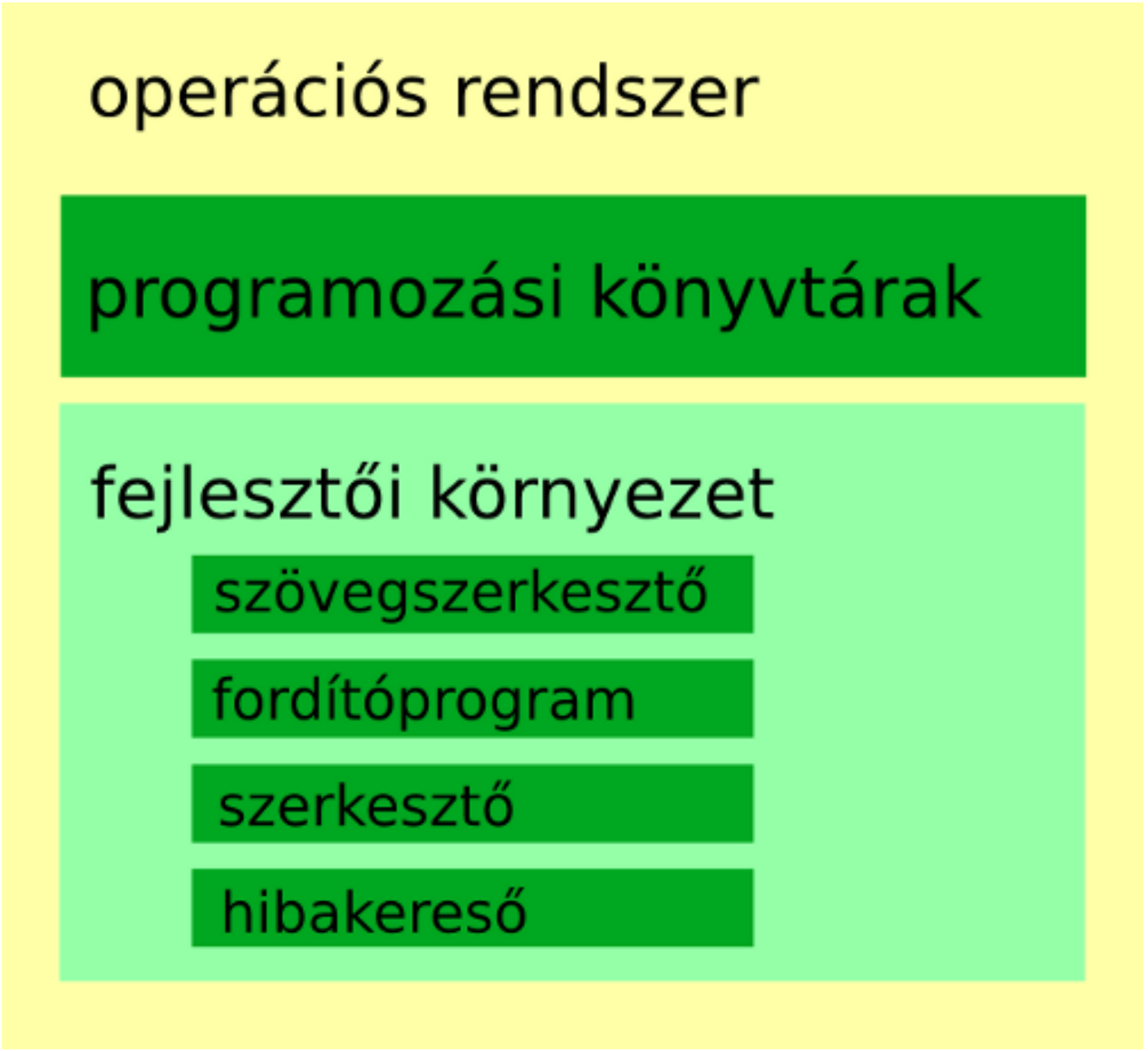
3. **Kódolás**: A kész terveket egy választott programozási nyelven kódoljuk. Vannak olyan szoftverek, amelyek a kódolást a terv alapján automatikusan elvégzik. Az elkészült *forráskódból* egy fordítóprogram hozza létre a futtatható *gépi kódú* programot.

4. **Tesztelés**: Minden program hibás. Az elkészült programot minimum két szempontból alaposan elemezni kell. (a) Minden esetben helyesen működik-e? (b) Hatékony-e? A hatékonyság három fő mérőszáma a tárhelyhasználat, a sebesség és a bonyolultság.

5. **Hibajavítás**: A felismert hibák kijavításához új specifikáció, új terv és újrakódolás lehet szükséges.

6. **Dokumentáció**: A programok nagyon nagyok. Ha egy év múlva is érteni szeretnénk működésüket, dokumentációt kell készítenünk. A dokumentáció két szintje a felhasználói és a fejlesztői dokumentáció.

PROGRAMOZÁSI KÖRNYEZETEK



- **Operációs rendszer**
- Kicsit általánosabban *platform*. A programokat mindig valamilyen platformra fejlesztjük. A választott platform befolyásolja, milyen eszközöket használhatunk, milyen korlátoknak kell megfelelnünk. A nyílt forráskódú fejlesztések jelentős része *multiplatform*, ami azt jelenti, hogy különböző operációs rendszerek alatt működő, lényegében azonos funkcionalitású változataik léteznek.
- **Programozási könyvtárak**
- A nagyobb fejlesztések általában előre gyártott komponenseket (is) használnak. A komponensek leggyakrabban programozási könyvtárakba csoportosítva érhetők el. Ahhoz, hogy felhasználhassuk programunkban a komponenseket, hozzá kell szerkesztenünk a megfelelő könyvtárakat.
- A programozási könyvtárak szerkesztése lehet statikus vagy dinamikus.
- **Fejlesztő környezet**
- A fejlesztői környezet legfontosabb elemei:
- **szövegszerkesztő** (source editor): ezzel készítjük el a program *forráskódját*, ami egy adott programozási nyelven írt egyszerű szöveges állomány
- **fordítóprogram** (compiler): a forráskódot lefordítja, jelzi a szintaktikai hibákat, ha pedig ilyen hiba nincs, elkészíti a *tárgykódot*
- **szerkesztő** (linker): a tárgykódból (vagy tárgykódokból), és esetlegesen programozási könyvtárakból szerkeszti össze a *futtatható programot*
- **hibakereső** (debugger): ha a fordítóprogram elhelyezett a tárgykódban bizonyos extra információkat, akkor a hibakereső lehetővé teszi, hogy a programot lépésenként futtassuk, és a lépések között ellenőrizhetjük a változók értékét

ALGORITMUSLEÍRÓ ESZKÖZÖK

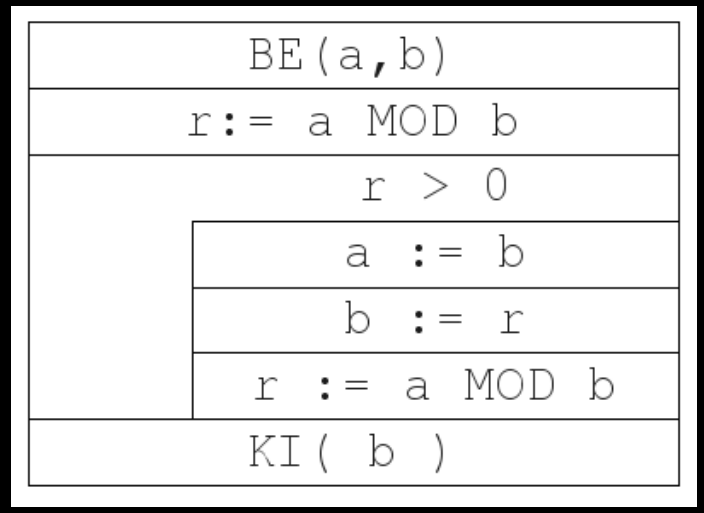
Egy 2300 éves algoritmikus probléma: határozzuk meg két pozitív egész legnagyobb közös osztóját!

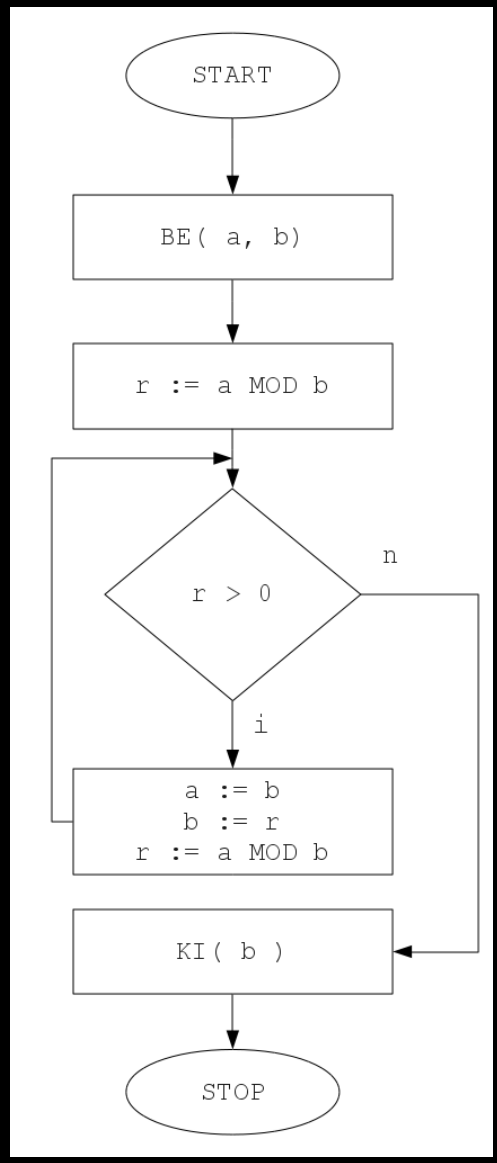
Példa

a = 360, b = 150 esetén a legnagyobb közös osztó (a,b) = 30.

AZ ALGORITMUS MONDATSZERŰ LEÍRÁSSAL

- BE(a, b) // Feltesszük, hogy "a" nagyobb...
- r := a MOD b
- Ciklus amíg r > 0
- a := b
- b := r
- r := a MOD b
- Ciklus vége
- KI(b)

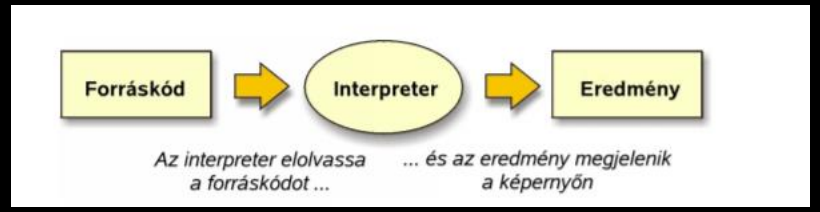




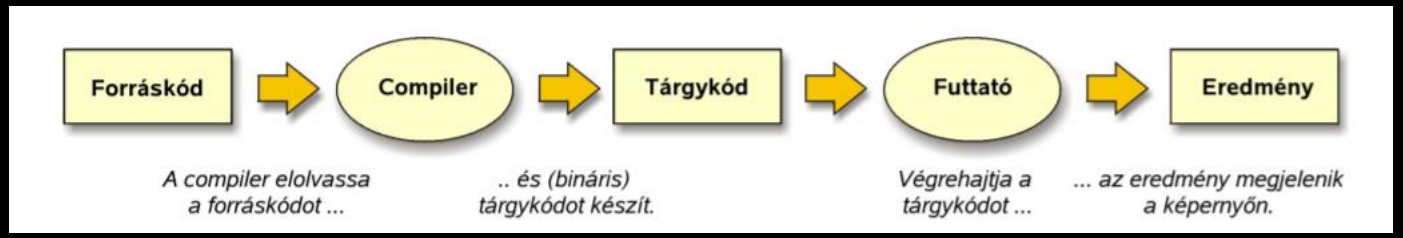
Hello, világ!

FORRÁSPROGRAM

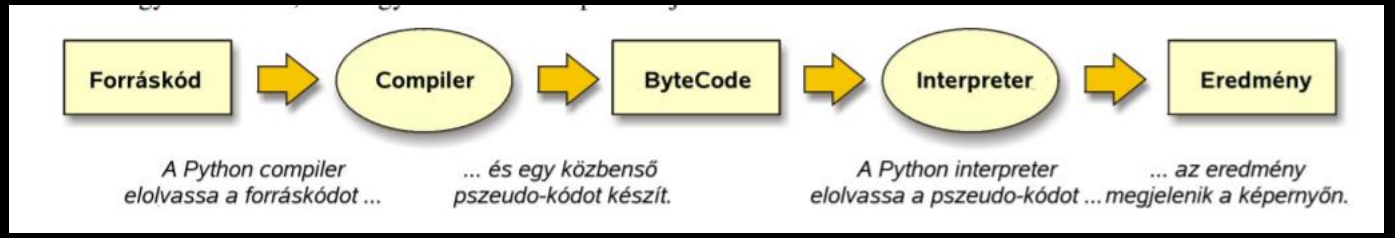
- Interpreter:
  néhány gépi nyelvű utasításra lefordítja, amiket azonnal végre is hajt, semmilyen tárgyprogram sem generálódik



- compilálás: A fordító program elolvassa a forrásprogram összes sorát és egy új kódot állít elő, amit tárgykódnak (object kód) hívunk



Python – modern nyelv:



PROGRAMFEJLESZTÉS - HIBAKERESÉS (« DEBUG »)

- Szintaxishibák: csak formai hiba, javítás és kész
- Szemantikai hibák: elméleti hiba, újragondolást igényel
- Végrehajtás közben fellépő hibák (run time error)