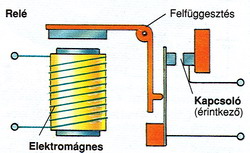
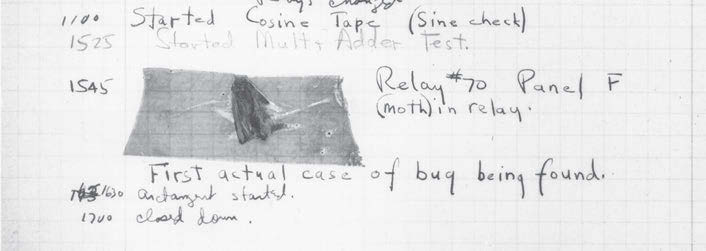
# A debugger használata

## Az első bug (hiba)

A legenda szerint az első programhibát a Harvard Egyetem Mark II számítógépében találták. Ez egy elektromechanikus számítógép volt, relékkel.



A hiba oka az volt, hogy egy rovar, pontosabban egy molylepke került az egyik relébe.



Ennek emlékére a programhibákat a mai napig bug-nak nevezik.

Ha egy programban nincsenek szintaktikai hibák, akkor le lehet fordítani, és el lehet indítani. Azonban még egyáltalán nem biztos, hogy hibátlanul működik.

A program működési hibáinak felderítésében segítenek a debugger programok. Ezek segítségével meg tudjuk állítani, és lépésenként tudjuk futtatni a programot, miközben nyomon követjük a változók értékeit.

A NetBeans is tartalmaz debuggert (hibakeresőt), a mai leckében ezzel fogunk megismerkedni.

## Példa: szótagszámoló program

Példaprogramunk szavakat olvas be, és kiírja, hány szótagból állnak. A programot üres szöveg megadásával lehet megállítani.

A megoldást az alapján készítettük el, hogy a szótagok száma megegyezik a magánhangzók számával. Ha jól működne a program, akkor így futna (célszerű egyszerű tesztadatokat választani):



Sajnos a program tartalmaz néhány hibát, amit ki kell javítanunk.

Nyisd meg a programot, és futtasd le! A következő eredményt kapod:



Először nézd meg a kódot, és gondold végig a program működését! A hibakeresés során a várt működést fogjuk összehasonlítani a tényleges működéssel.

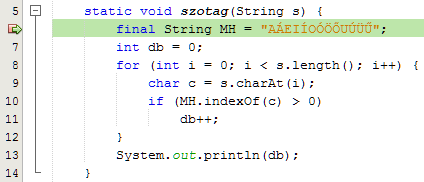
Mivel a szótagok számlálásával van probléma, valószínűleg a szotag() metódusban lesznek a hibák.

A hibakereséshez először meg kell adni egy töréspontot, ahol majd megáll a program. Legyen ez a 6-os sor!

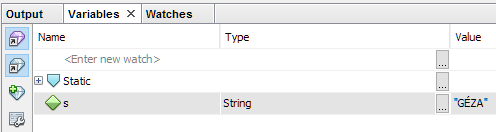
Kattints a 6-os sorszámra! A töréspontot rózsaszín háttér és a sorszám helyén megjelenő négyzet jelzi.

Ezután indítsd el a programot a Debug Project gomb vagy a Ctrl+F5 megnyomásával! Írd be a Géza szót, és nyomj Entert!

A program a töréspont végrehajtása előtt meg fog állni. Ekkor a sor háttérszíne zöldre vált, és megjelenik egy nyíl a sor előtt.



Az ablak alsó részén az Output fül mellett megjelenik a Variables és a Watches fül is. Válts most a Variables fülre! Itt fogod látni a változók aktuális értékeit. Most a paraméterként kapott s értékét (és típusát) láthatod. Ezzel nincs probléma.



Ezután lépésenként fogjuk futtatni a programot az F8 megnyomásával. Használhatod a megjelenő Debug eszköztár Step over gombját is.

Nyomd meg kétszer az F8-at, és figyeld meg, hogy az MH és a db változók is létrejöttek és értéket kapnak!

Léptess végig egyszer a for cikluson, és figyeld meg, mi történik! Mivel a G nem magánhangzó, nem megy bele az if utasításba, így a db értéke nem változik.

Menj végig még egyszer a for cikluson, és figyeld meg, hogy az é betűt sem veszi magánhangzónak! Miért? Nézd meg az MH változó tartalmát!

Igen, az MH-ból kimaradt az É betű. Állítsd meg a programot (Shift+F5 vagy a rózsaszín négyzet a Debug eszköztáron), és javítsd ki a hibát!

Próbáld ki újra a programot (F6-tal indítva, debugger nélkül), és figyeld meg, hogy még mindig nem teljesen jó az eredmény! Az A betűt még mindig nem veszi figyelembe, pedig az szerepel az MH változóban:



Most a 10. sort szeretnénk alaposabban megvizsgálni, mert itt dönti el a program, hogy egy karakter magánhangzó-e. Ezért kapcsold ki a töréspontot a 6. sorban (kattints a négyzetre), és készíts egy másik töréspontot a 10. sorba!

Indítsd el ismét a programot, de most a Debug Project gombbal (Ctrl+F5), és írd be a Géza szót! Amikor a program megáll a töréspontnál, akkor válts alul a Variables ablakra, és nézd meg a változók értékeit!

Most nem egyesével fogunk lépkedni, hanem futtatjuk a programot a következő töréspontig az F5 vagy Continue gomb megnyomásával. A következő töréspont ugyanitt lesz, csak akkor már az É betű lesz a c változóban.

Nyomd meg még kétszer az F5 billentyűt! Ekkor a töréspontnál az A karakter lesz a c változóban.

Most az F8 megnyomásakor azt várjuk, hogy menjen bele az if utasításba, és növelje meg a db változó értékét. De azt tapasztaljuk, hogy ez nem történik meg! Vajon miért?

Vidd a kurzort az IndexOf metódusba, és nyomd meg a Ctrl+szóköz billentyűket! Ennek hatására megjelenik egy rövid leírás a metódusról. Olvasd el! Milyen értéket ad vissza a metódus, ha nem találja a karaktert a stringben?

Igen, -1-et és nem nullát! Állítsd meg a hibakeresést, javítsd ki a hibát, és próbáld ki újra a programot normál futtatással! Most már jól fog működni.

Távolítsd el a töréspontot a 10. sorból! (Ezt egyébként hibakeresés közben is meg lehet tenni.)

## A hibakeresés lehetőségei

Az alábbi táblázat a hibakeresés közben használható fontosabb lehetőségeket foglalja össze:

|  |  |  |
| --- | --- | --- |
| Gomb | Billentyű | Leírás |
| Step Over | F8 | Egy utasítás végrehajtása úgy, hogy nem lép bele a metódusokba. |
| Step Into | F7 | Egy utasítás végrehajtása úgy, hogy híváskor belelép a metódusokba. |
| Step Out | Ctrl+F7 | Kilépés a metódusokból a hívó metódusba. |
| Continue | F5 | A program folytatása a következő töréspontig. |
| Finish Debugger Session | Shift+F5 | Hibakeresés vége, a program megállítása. |

## Feladat

Nyisd meg a *megfordit* nevű projektet! A programnak szövegeket kell beolvasnia és fordítva kiírnia. A végét most is az üres szöveg megadása jelzi.

Próbáld ki a programot, és figyeld meg a hibákat!

Keresd meg a hibák okait és javítsd ki őket! Használd a hibakeresőt!

Hány hiba volt? Írd ide: 2