# ’HIBAKERESÉS’ GYAKORLAT
# *
# PARTITÚRÁJA

*Szlávi Péter*

2012

## VÁZLAT

# *Bevezetés*

## A gyakorlat „filozófiája”

Néhány feladathoz tartozó, különféle hibákat tartalmazó programokban kell hibát keresni, helyesebben megtalálni.

A leggyakoribb hibajelenségek:

- futási hiba
- rossz (hiányos, helytelen…) eredmény
- kivárhatatlan (vagy gyanúsan lassú) futás
- …

Az alábbiakban kifejezetten a **kódolás** során gyakorta elkövetetett hibát sorolunk föl, amelyekre „ki kell élesedjen” a szemünk:

- **Inicializálatlan**ul hagyott **változó**. Mindenféle hiba forrása lehet.
- Lexikális elem **elírása**. Az elírások sokszor szintaktikus hibákként jelentkeznek. Szerencsétlen esetben az elírás utáni lexikális elem is „értelmes”, így a fordító „tud vele mit kezdeni”. Ezért nehéz a lokalizálása. „Közeli nevű” változó/konstans/eljárás/függvény, sőt azonos nevű, de lokálisan nem deklarált és globálisan létező adat is okozhat hibát…
- **Meggondolatlan** (illegális) **segédváltozó** (pl. ciklusváltozó) **használat**. Gyakori következmény a végtelen vagy „hirtelen” véget érő ciklus.
- A formális paraméter **nem megfelelő hozzáférési jog**ának megadása. Ez szerencsésebb esetben szintaktikai hibát okoz. A mi esetünkben –ha ilyen hiba egyáltalán van–, akkor rejtőzködik, nehéz rábukkanni. Misztikus „eredménytelenség” a nehezen lokalizálható alprogramban.
- **Nem megfelelő**en „tág” **paramétertípus**. Gyakori hibajelenség: a típus értéktartományán kívül esés.
- **Fordítási konstansok nem konzekvens** (nem azonos) **alkalmazásai**. Ilyenek gyakran szerepelnek tömbméret-megadásban és a tömbbel kapcsolatos transzformációkban. Gyakori hibajelenség: az indextúllépés.
- **Összetett utasítás helytelen** (többnyire elmaradó) **zárójelezése**. Mindenféle hibák forrása.

## A fejlesztési környezet adta lehetőségek

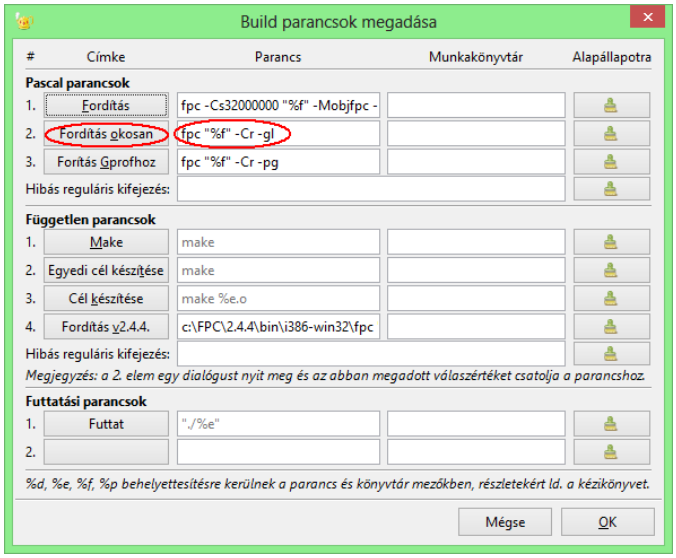
A hibakeresést célszerűen olyan fejlesztői környezetben érdemes (könnyű) végezni, amelynek vannak hibakeresést támogató szolgáltatásai: **lépésenkénti végrehajtás**, **töréspont elhelyezés**, jellegzetes **adatok értékeinek megjelenítése** stb. Ilyen környezet a Pascal esetében a Free Pascal IDE.[1] Ennél a környezetnél könnyedén szabályozható a fordítás „szigorúsága” is. Éljünk ezzel a lehetőséggel! (L. az `Option` menü ’`Range check`’ beállítást!)

---

[1] Ha esetleg ajánlhatom egy korábbi változat hordozhatóvá tett változatát, amely innen letölthető: http://people.inf.elte.hu/szlavi/FP_portable/FP_portable.zip.

Vagy keresse meg a gépére installált FreePascal natív IDE-jét (fp.exe), és indítsa el. Ez „ilyenszerű” (nem biztos, hogy pontosan ez) elérési úttal szokott rendelkezni: C:\FPC\3.0.0\bin\i386-win32\fp.exe.

A Geany nem nyújt segítséget a nyomkövetéshez. Esetében saját állapotkiíró és futást felfüggesztő rutint/rutinokat kell írnunk. Ámbár rávehető a Geany is arra, hogy némi segítséget nyújtson a hibakeresésben. Módosítsuk a fordítási parancsot így: fpc "%e" **–Cr –gl**. (L. az 1. ábrát.) Így a 'Range check' hibafigyelés be lesz kapcsolva, és a „szokásos" futási hibakódok (pl. 201 – indextúllépéskor) mellett a hiba helye is (a forrásfájl hányadik sorában következett be) megjelenik.




**1. ábra:** A segítőkészebb fordítás beállítása Geany-ben.

A **futási hibák kódjai**ról összeállítást itt találhat:
http://www.freepascal.org/docs-html/user/userap4.html.

## *Anyagok a gyakorlathoz, és teendők a gyakorlaton*

Az elkövetkezőkben 3 feladathoz tartozó, többféle hibát tartalmazó megoldásváltozatban kell hibát keresni és javítani.

### Anyagok

Ezeket egyetlen csomagban lehet letölteni:

http://people.inf.elte.hu/szlavi/PrM1felev/Hibakereses/LetoltendoGyak_FP.zip

A feladatokhoz tartozó változatok:

- HIBA0_**x**.PAS (**x**:1,2A,2B,3,4A,4B,5,6) – az első feladat forráskódjai
- HIBA0.exe – az első feladat helyes kódja
- HIBA1_**x**.PAS (**x**:1,2,3,4) – a második feladat forráskódjai
- HIBA1.exe – a második feladat helyes kódja
- HIBA2_**x**.PAS (**x**:1,2,3,4,5) – a harmadik feladat forráskódjai
- HIBA2.exe – a harmadik feladat helyes kódja

Az IDE használatához némi tanács olvasható a http://people.inf.elte.hu/szlavi/ProgModsz/Tanacsok.pdf dokumentum „Apróságok" fejezetében (3. oldal aljától).

## Teendők

Mindenekelőtt állítsa be a fejlesztői környezetet! (Fordítási opciók.) Ha a „natív” FP IDE-t használja, akkor azt töltse le, csomagolja ki egy megfelelő könyvtárba, majd indítsa el az FP_Indit.bat batch-programmal. Ha Geany-t használ, állítsa be az „előzékeny” fordítást, és azzal fordítson később!

Az egyes változatokkal tegye ezt:

- Értse meg a megoldás „logikáját”!
- Fordítsa le és futtassa az adott változatot!
- Hasonlítsa össze a kapott eredményeket (feltéve, hogy egyáltalán született valamilyen értelmezhető kimenet) a helyes program eredményeivel! Ha kell, futtathatja a helyes változatot.
- Keresse a hiba okát és a helyét; majd javítsa ki!

## *Első feladat*

A **majdnem-barátságos számok** kigyűjtése N-ig, pontosabban, amíg a pár kisebbik tagja<N. A „majdnem-ség”: az 1-et nem számítjuk az osztók közé (ahogy magát a számot úgy sem kell). Ebben tér el a matematikában meghonosodott barátságos számok definíciójától. Ennek a majdnem-ségnek a következménye, hogy némileg nagyobb lesz az ilyen számok gyakorisága.

### Egy példa teszteset:

**Bemenet:**

```
N: 9999
```

**Kimenet:**

```
1:   48,   75
2:  140,  195
3:  320,  441 [441 négyzetszám; így a 21-et 2-szer számoljuk osztóként!]
4: 1050, 1925
5: 1575, 1648
6: 2024, 2295
7: 5775, 6128
8: 8892,16587
9: 9504,20735
```

## *Második feladat*

„Valódi tökéletes számok” kigyűjtése az előbbi, „majdnem-barátságos számokat” kigyűjtő programból eredeztetve.

### Egy példa teszteset:

**Bemenet:**

```
  N: 9999
```

**Kimenet:**

```
  1:    6
  2:   28
  3:  496
  4: 8128
```

## *Harmadik feladat*

Generáljunk egy sorozatot a következő módon:

$x_0$:=N → $x_1$:=OsztóÖsszeg($x_0$) → $x_2$:=OsztóÖsszeg($x_1$) → … → $x_i$:=OsztóÖsszeg($x_{i-1}$)

Mindezt addig végezze, amíg $x_i>1$ és $x_i \notin \{x_1,x_2,...x_{i-1}\}$, és ellenőrzésképpen még az utolsó utánit ($x_{i+1}$) illessze a sorozathoz (amelyre már nem teljesül az előbbi feltétel).

### Négy példa teszteset:

| 1. teszteset | 2. teszteset | 3. teszteset | 4. teszteset |
|---|---|---|---|
| **Bemenet:**<br>12 | 10 | 6 | 220 |
| **Kimenet:**<br>1: 12<br>2: 16<br>3: 15<br>4: 9<br>5: 4<br>6: 3<br>7: 1 | 1: 10<br>2: 8<br>3: 7<br>4: 1 | 1: 6<br>2: 6 | 1: 220<br>2: 284<br>3: 220 |