1. Násobenie prirodzených čísel s jednou globálnou premennou

Pri tejto úlohe som najprv premýšlal o zmysle života, potom po dvoch dňoch som skúsil implementovať ideu zakódovania nejakých čísel do prvočísel, ako sme to robili na cviku.

Pôvodne som zvolil zakódovanie do 2^A3^B, kde A, B sú dve načítané čísla zo štandardného vstupu. To som však musel neskôr zmeniť z pomerne praktického dôvodu. Prvý vstup A som tak zakódoval do 3^A klasickým násobením 3-kou pri vynáraní z rekurzie. Druhý vstup B som zakódoval podobne, avšak do 5^B.

Vďaka tomuto kódovaniu 3^A5^B sa vieme spätne dostať k pôvodným hodnotám A, B. Táto idea sa dá využiť aj pri získaní výsledku, teda súčinu A, B. Pridajme do kódovania teda ďalšie prvočíslo 2, t.j. $3^A5^B2^0$. Idea je taká, že simulujeme vnorený cyklus:

```
for i:= 0; i < A; i++
    for j:=0; j < B; j++
        a = a * 2
```

- pre každé delenie 3-kou začnime deliť 5-kou
- pri každom delení 5-kou si naše zakódované číslo vynásobme 2-kou
- takto postupne zvyšujeme hodnotu exponentu pri 2-ke $A \times B$ krát
- vo výsledku máme číslo zakódované ako $3^A5^B2^R$, kde R je náš súčin A a B

Teraz už len stačí podeliť zakódované číslo 2-kou kým sa dá a počet týchto delení je náš výsledok.

Pôvodne som kódoval ako $2^A3^B5^R$, čo mohlo veľmi rýchlo spôsobiť (aj spôsobilo) pretečenie pri kódovaní výsledku R, tak som ho prehodil do 2-ky. Ani to však veľmi nepomohlo, už pri násobení 7×7 program pretečie. Po diskusii s vyučujúcim som skúsil vytvoriť alternatívny program v Pythone, tam to zvládne cca 13×13. Nemáme zasahovať do deklarácie premennej a, t.j. k stabilnejším výsledkom sa asi pri tejto stratégii nedopracujem.

2. Súčet mnohých celých čísel

V tejto úlohe využívam úplne jednoduchú ideu:

- Načítam číslo. Ak je nenulové:
  - Ak je načítané číslo $a > 0$, pri vnáraní do rekurzie sa znižuje, až kým nie je nulové. Pri vynáraní sa naopak zvyšuje.
  - Ak je načítané číslo $a < 0$, pri vnáraní do rekurzie sa zvyšuje, až kým nie je opäť nulové. Pri vynáraní sa naopak znižuje.
- Inak sa začne vynárať a k zadanej 0 korektne pripočíta všetky zadané čísla.

3. GO tour 1

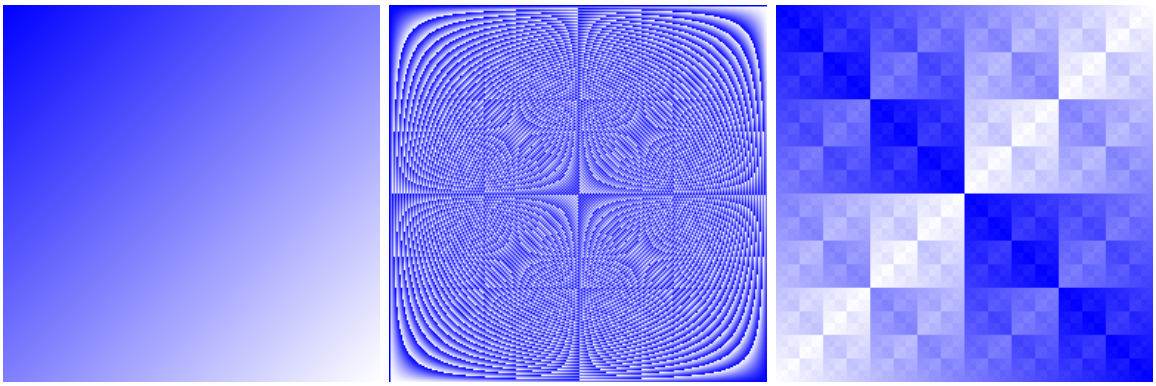
Exercise: Loops and Functions

Zadanie je asi vcelku jasné, trebalo len otestovať rôzne varianty parametrov, kedy budu výsledky danej funkcie čo najviac porovnateľné so štandardnou *math* knižnicou. Vyskúšal som odmocniť prvých $n = 20$ prirodzených čísel a najviac zhodných výsledkov (15) mi dalo štartovné $z = 0.1$ a $delta = 0.000001$ (prípustná odchýlka).

Exercise: Slices

Dúfam, že som pochopil zadanie, lebo sa mi to zdalo až príliš jednoduché, t.j. vytvoriť dvojrozmerné pole a vrátiť ho.

Vytvára to však pekné obrázky (v online editore):



4. Veľké kombinačné číslo

Pokúsil som sa naprogramovať algoritmus pomocou idey na tomto fóre. Podobne ako na cvičeniach som šiel rovno na *big.Int a využíval jeho metódy. Samotný algoritmus je pomerne jednoduchý, počíta len jeden, $n-tý$ riadok a využíva fakt, že riadok Pascalovho trojuholníka nám stačí počítať len do hľadaného $\binom{n}{k}$.

V cykle znižujeme teda n (v premennej i), až po $k+1$, a zároveň zvyšujeme deliteľa j. n vynásobíme s $(n-1)$ a vydelíme j. Vypočítaný medzivýsledok potom násobíme s $(n-2)$, atď.

Výsledky v porovnaní s vyučujúcim nie sú svetaborné, ale ani nie najhoršie (i5, 8GB RAM):

```
comb(50, 25):
- computing time: 14.94mcs
- first 10 digits: 1264106064
- total digit count: 15

comb(100, 50):
- computing time: 21.501mcs
- first 10 digits: 1008913445
- total digit count: 30

comb(1000, 500):
- computing time: 256.816mcs
- first 10 digits: 2702882409
- total digit count: 300

comb(10000, 5000):
- computing time: 18.998531ms
- first 10 digits: 1591790263
- total digit count: 3009

comb(50000, 25000):
- computing time: 447.624109ms
- first 10 digits: 1127810378
- total digit count: 15050

comb(100000, 50000):
- computing time: 1.769817275s
- first 10 digits: 2520608368
- total digit count: 30101

comb(200000, 100000):
- computing time: 6.966282848s
- first 10 digits: 1780562887
- total digit count: 60204

comb(500000, 250000):
- computing time: 43.291136094s
- first 10 digits: 1122759743
- total digit count: 150513

comb(1000000, 500000):
- computing time: 2m54.703846238s
- first 10 digits: 7899578772
- total digit count: 301027
```