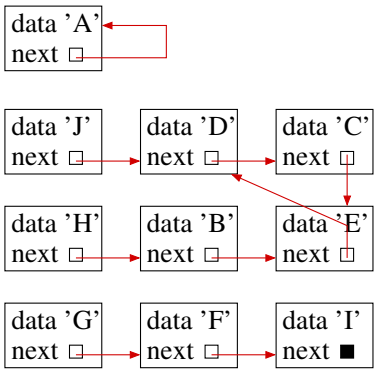
Programovanie (1) v C/C++ 2025/26
Cvičenia 8, príklad 6, bonus
Cykly

Priložená kostra obsahuje štruktúru `node` pre spájaný zoznam, v ktorom sú ako dáta znaky (`char`) ako aj funkciu `main`, v ktorej je hotové načítanie a výpis. **Hotové časti programu nemodifikujte.**
Vašou úlohou je naprogramovať funkciu `zisti`. Táto funkcia dostane smerník na záznam typu `node`. Ak by sme z tohto záznamu postupovali ďalej pomocou smerníkov `next` na ďalšie uzly spájaného zoznamu, môžu nastať dve situácie. Buď po čase prídeme na uzol, ktorý má `next` nastavené na `NULL`, alebo je zoznam "zacyklený", takže by sme po smerníkoch chodili do nekonečna. Pre nezacyklené zoznamy má funkcia vrátiť počet uzlov, ktoré navštívime, kým prídeme na koniec zoznamu a pre zacyklené má vrátiť hodnotu -1. Napríklad ak v situácii na obrázku `zisti` dostane smerník na uzol so znakom `F`, má vrátiť 2, lebo z toho uzlu vieme ešte navštíviť uzol `I` a potom zoznam končí (nevšímame si uzol `G`, lebo do neho sa z uzla `F` nedostaneme). Ak funkcia dostane smerník na uzol `H`, funkcia má vrátiť -1, lebo ak by sme pokračovali po smerníkoch, dostaneme sa do uzlov `B`, `E`, `D`, `C`, `E`, `D`, `C`, ... a nikdy by sme neprišli na hodnotu `NULL`.



Algoritmus: Po spájanom zozname sa budeme pohybovať dvoma smerníkmi x a y. V každej iterácii cyklu sa smerník x pohne o jeden krok, smerník y o dva kroky. Smerník y bude teda "popredu" a ak je zoznam ukončený smerníkom s hodnotou `NULL`, po určitom počte krokov na túto hodnotu narazí a vieme, že zoznam nie je zacyklený. Ak sme počítali počet posunov smerníka y, poznáme aj dĺžku zoznamu, ktorú môžeme vrátiť.

Ak po niekoľkých krokoch nastane situácia, že x a y sa rovnajú, vieme, že zoznam zacyklený a môžeme vrátiť hodnotu -1. Ako ukážeme nižšie, toto sú jediné dve situácie, ktoré potrebujeme uvažovať. Ak by sme v našom príklade začali vo vrchole `F`, po prvom kroku bude x ukazovať na `B` a y na `E`. Po ďalšom kroku budú ukazovať na `E` a `C`, a po ešte ďalšom sa stretnú na uzle `D`. Vo vašom programe **implementujte tento postup**.

Správnosť algoritmu: Ukážme, že tento algoritmus naozaj funguje. V prvom rade, ak je zoznam nezacyklený, y vždy popredu pred x a nikdy sa nebudú rovnať, takže algoritmus naozaj správne skončí tým, že y bude `NULL`.

Zostáva nám teda ukázať, že ak je zoznam zacyklený, po určitom počte krokov dôjde k situácii, že x a y sa rovnajú. Ako vidíme na obrázku, na začiatku zoznamu môže byť niekoľko uzlov, ktoré nepatria do cyklu (ak sme začali v uzle `H`, sú to uzly `H` a `B`). Ale po určitom počte krokov sa aj x aj y dostanú do cyklu a budú tam krúžiť. V okamihu, keď pomalší smerník x dosiahne cyklus, je rýchlejší smerník y už niekde v cykle a nech k je hodnota taká, že keby sa y posunulo o k uzlov dopredu, došlo by na pozíciu x. Môžeme teda situáciu chápať tak, že smerník y je o k uzlov pozadu za x. Smerník y je ale rýchlejší a v každom kroku algoritmu zmenší náskok smerníka x o jedna. Po k krokoch ho teda dobehne a budú sa rovnať.

Poznámka: Možno vás napadnú aj iné spôsoby, ako by sme túto úlohu mohli riešiť. Tento algoritmus však má niekoľko výhod. Nemodifikuje uzly, takže sa dajú použiť na ďalšie výpočty. Nepoužíva ani žiadne pomocné polia a podobne, jediná pamäť navyše sú dva smerníky a počítadlo krokov. A ak je dĺžka zoznamu vrátane prípadného cyklu n, spraví najviac $3n$ posunov smerníkov, takže je aj efektívny.

Vstup a výstup: Vstup a výstup sú už naprogramované. Na prvom riadku vstupu je počet uzlov n. Na každom ďalšom riadku je popísaný jeden uzol: najskôr znak uložený ako dáta (bude to vždy nebiely znak) a potom poradové číslo uzla, na ktorý má aktuálny uzol ukazovať. Ak nemá ukazovať na žiaden uzol, číslo je -1. Program na výstupe vypíše pre každý uzol v rovnakom poradí ako na vstupe najskôr dáta uložené v uzle a potom výsledok volania funkcie `zisti`.

Príklad vstupu:

```
10
A 0
B 5
C 5
D 2
G 7
E 3
I -1
F 6
J 3
H 1
```

Príklad výstupu:

```
A: -1
B: -1
C: -1
D: -1
G: 3
E: -1
I: 1
F: 2
J: -1
H: -1
```