Univerzita Karlova
Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

ÚVOD DO PROGRAMOVÁNÍ

Nalezení samohlásek v zadaném textu a jejich zvýraznění umístěním do závorek

dokumentace

Lukáš Nekola
2. ročník, NKARTGD
Humpolec 2022

Obsah

1. Zadání

Úkolem je vytvoření programu, resp. aplikace, která dokáže v uživatelem zadaném textu vyhledat samohlásky a náležitě je označit umístěním do závorek. Existence několika možných algoritmů podněcuje možnost volby. Pro snazší a jednoznačnější výsledek bude dále vytvořeno počítadlo samohlásek v textu.

2. Rozbor problému

V této části bych rád popsal a rozebral samotnou problematiku. Již nyní je zřejmé, že se bude jednat především o práci a manipulaci s textovým řetězcem a jednotlivými znaky v něm obsaženými. Všechny možné způsoby se opírají o definici konstanty textového řetězce možných samohlásek. Tento řetězec musí být obohacen o velká písmena.

3. Existující algoritmy

Popis existujících algoritmů pro vyhledávání v textovém řetězci bude velmi problematickou záležitostí z důvodu neexistence předepsaných postupů pro vyhledávání samohlásek. Nicméně dle webového portálu GeeksforGeeks (2020) existují dva základní způsoby nalezení pozice, resp. indexu, samohlásek. První způsob se zabývá procházení textového řetězce či jiné datové struktury pomocí cyklu a následné vrácení pozice či samotné samohlásky. Druhý způsob se zaobírá výčtem objektů z určité kolekce hodnot (funkce *enumerate*), kterým může být například seznam či textový řetězec. Následně jsou za pomocí podmínky vybrány takové objekty, resp. prvky, které mají podobu samohlásek.

4. Popis zvoleného algoritmu

Na počátku můžeme napsat, že rozdělíme na 2 části týkající se počítání samohlásek a označení samohlásek v textovém řetězci. Nicméně před samotným započetím je nutné definovat 2 konstanty textových řetězců se samohláskami, kdy jedna bude ve formátu verzálek a druhá minusek.

První část algoritmu se zabývá zjištěním počtu samohlásek v zadaném textu uživatelem. Funguje na principu procházení jednotlivých znaků daného textového řetězce. Pokud algoritmus narazí na podobu znaku, která je definovaná v jedné z konstant (samohláska), přičte hodnotu 1 k dosavadnímu počtu samohlásek.

Druhý algoritmus pro vyznačení samohlásek v textu funguje opět na principu procházení jednotlivých znaků textového řetězce v konečném cyklu. Pokud vybraný znak není definovaný v jedné z konstant, resp. nejedná se o samohlásku, je vypsán do nového řetězce. Pokud se jedná o samohlásku, text je nejdříve naformátován do podoby, kdy je samotný znak obalen do závorek. Následně je opět zapsán na poslední pozici již zmíněného nového textového řetězce. Na konci algoritmu jsou jednotlivé nové proměnné vypsány.

5. Struktura programu (datové struktury, metody)

Samotný program je sepsán na pouhých 23 řádků včetně komentářů. Musím podotknout, že vzhledem k nízké náročnosti nebylo nutné definovat metody. Strukturu programu můžeme dělit na následující části:

a) První část se zabývá vstupem textového řetězce do samotného programu. Tento textový řetězec není nutné kontrolovat a je dovoleno využívat všechny znaky.
b) Pro následné provedení algoritmů byly definovány 2 konstanty textových řetězců s názvy *VOWELSMALL* a *VOWELBIG.* Jak napovídají názvy, jedná se o definici samohlásek ve verzi minusek a verzálek.
c) Následně je provedena část algoritmu pro výpočet počtu samohlásek v řetězci, jejíž fungování je sepsáno ve 4. kapitole. Jedná se tedy o *FOR* cyklus, který prochází jednotlivé znaky textového řetězce. Pokud se jedná o samohlásku definovanou, přičte se hodnota 1 do proměnné typu *integer*. Původně je tato proměnná o hodnotě 0 definována před samotným započetím cyklu.
d) Čtvrtá část programu obsahuje sepsaný druhý algoritmus ze 4. kapitoly. Nejdříve je vytvořena proměnná *newStr* typu *string*, která je prázdnou množinou. Do této proměnné se postupně zapisují jednotlivé znaky na základě procházení textové řetězce *For* cyklem. Pokud je současný znak samohláskou, zapíše se na konec textového řetězce *newStr* obalený závorkou. Pokud se nejedná o samohlásku, zapíše se bez úprav.
e) V poslední části jsou vzniklé proměnné vytisknuty do konzole a výsledek je náležitě popsán pro lepší pochopení.

6. Pseudokód

funkce *CalculateVowels (textStr, VOWELSMALL, VOWELBIG)*
    inicializace proměnné countVowel s počáteční hodnotou počtu samohlásek 0
    cyklus *procházení jednotlivých znaků řetězce textStr*
        pokud *je znak součástí z některé z konstant definovaných níže*
            zvýšení hodnoty proměnné countVowel o 1
    vrácení *proměnné countVowel*

VOWELSMALL *= konstanta definující samohlásky ve tvaru malých písmen*
VOWELBIG *= konstanty definující samohlásky ve tvaru velkých písmen*
textStr *= vložení libovolného textu uživatelem*

newStr *= prázdný textový řetězec*
cyklus *procházení jednotlivých znaků řetězce textStr*
    pokud *je znak součástí z některé z konstant*
        přidání znaku do řetězce newStr, kdy znak je obalený závorkami
    pokud *znak není součástí některé z konstant*
        zápis aktuálního znaku na poslední pozici řetězce newStr

pokud *délka vstupního řetězce je větší než 0*
    výpis *počtu samohlásek v řetězci na základě definované funkce*
    výpis *upraveného textového řetězce s vyznačenými samohláskami*
pokud *je délka vstupního řetězce*
    výpis *informace o prázdném řetězci*

7. Popis vstupních/výstupních dat

Za vstupní data považujeme textový řetězec, který zadá uživatel do konzole, resp. příkazové řádky. Prvním výstupem je celočíselná hodnota odkazující na celkový počet samohlásek v uživateli zadaném textovém řetězci. Druhý výstup zvýrazňuje jednotlivé samohlásky v textu pomocí závorek. Vstup a výstup je zobrazen v textové příloze tohoto dokumentu.

8. Problematická místa

Za problematické místo můžeme označit existenci dvou cyklů, jejich počet iterací je závislý na délce řetězce.

9. Možná vylepšení

Za možné vylepšení bychom mohli považovat například nalezení méně náročného způsobu výběru samohlásek v textu, jelikož samotné cykly mohou být časově náročné při delším vstupním řetězci. Náročnost průběhu by mohla být omezena definicí maximální

délky vstupního řetězce. Nicméně tento fakt je použitelný pouze ve specifických problematikách. Nemožnost vstupu číselných hodnot můžeme považovat za obdobnou problematiku. Za poslední možné vylepšení považuji převedení všech znaků textového řetězce na malá písmena, což by podněcovalo použití snazších podmínek bez deklarace a inicializace 2. konstanty.

10. Zdroje

GeeksforGeeks (2020): *Python | Vowel indices in String*
https://www.geeksforgeeks.org/python-vowel-indices-in-string/ (cit.21.01.2022)