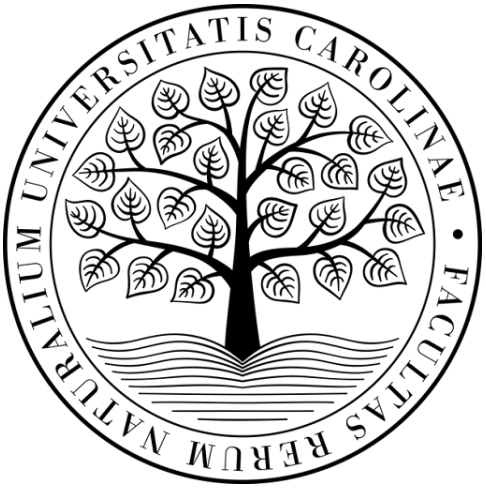
Univerzita Karlova

Prírodovedecká fakulta



Daniel Kalakay

2. ročník študijného programu Sociálna geografia a geoinformatika

Výpočet četností znaků v textu.

Dokumentácia k skúškovému programu na predmet:

Úvod do programování

Akademický rok 2025/2026

Praha 20. 1. 2026

Obsah

Úvod a stručné zadanie

Táto práca sa zaoberá návrhom, vypracovaním a následným zdokumentovaním programu pre výpočet absolútnej a relatívnej početnosti znakov vo vstupnom texte zadaného používateľom. Celý program je zostrojený v jazyku Python a v prostredí PyCharm, využíva princípy objektovo orientovaného programovania (OOP).

Cieľom práce bolo vytvoriť program, ktorý je prehľadný, rozšíriteľný a používateľsky jednoduchý a zrozumiteľný. Program podľa zadania analyzuje text na základe predom určených písmen a znakov slovenskej abecedy. Filtruje nepodporované znaky bez zbytočného výpisu a výsledky zobrazuje od najpočetnejšieho znaku k menej početným.

Presné zadanie úlohy

Pre vstupný text zahrňujúci písmena "A-Ž", "a-ž", číslovky "0-9" a špeciálne znaky ",.!?;" oddelené medzerami vypočítajte absolútnu a relatívnu početnosť jednotlivých znakov. Výsledok vo formáte znak / absolútna početnosť / relatívna početnosť vypíšte zostupne podľa hodnôt absolútnych početností. Ak sa v texte budú nachádzať nepodporované znaky, ignorujte ich.

Analýza úlohy

Úlohu je možné jednoduchšie preformátovať do formy analýzy textového reťazca v ktorom je nutné zapracovať nasledovné parametre s cieľom:

- rozlíšiť podporované znaky od nepodporovaných,
- zosumarizovať finálnu početnosť znaku,
- vypočítať relatívnu početnosť znaku vzhľadom k celému textu,
- zoradiť následné výsledky zostupne,
- vypísať výsledky.

Z obecného zadania pre riešenie každého príkladu je možné využiť OOP. Najvýhodnejšie je zabaliť celý proces do jednej triedy, ktorá následne ponesie zodpovednosť za spracovanie všetkých dát (či už to je výpočet početností, uchovávanie a usporiadanie týchto dát a následné vypísanie výsledkov). Výhodou tohto prístupu je menšie zaťaženie celkového procesu, jednoduchšia manipulácia s dátami pri prípadnom rozširovaní programu a globalizácia použitých premenných.

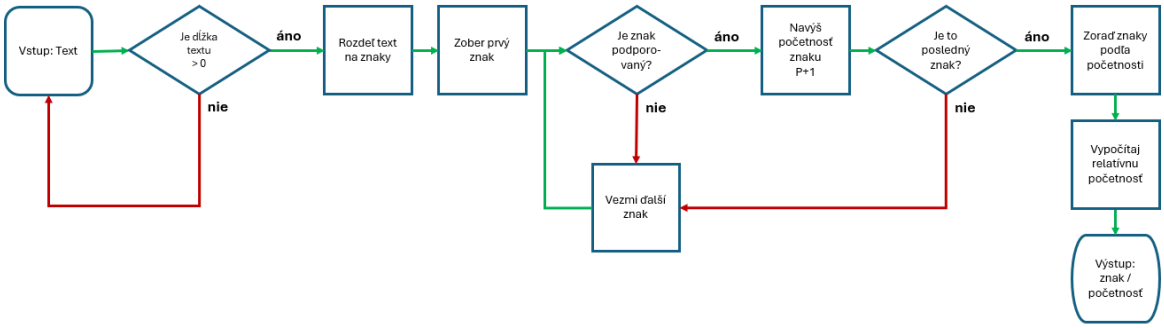
Voľba algoritmu

Návrh objektu

Program je navrhnutý ako modulárny systém založený na triede Analyzátor, ktorá reprezentuje celý výpočtový proces a zabezpečuje jeho rozdelenie do jednotlivých funkčných častí, ako sú napríklad hlavná riadiaca funkcia a mechanizmus výpisu výsledkov. Trieda obsahuje: prázdnu premennú text, do ktorej bude načítaný vstupný text zadaný používateľom, množinu všetkých podporovaných znakov podľa zadania, slovník pre ukladanie absolútnej početnosti jednotlivých znakov a premennú uchovávajúcu celkový počet spracovaných znakov. Jednotlivé časti odpovedajú logickým krokom navrhnutých v časti analýza úlohy.

Použité postupy a algoritmy

Vložený text program spracováva znak po znaku pomocou sekvenčného priechodu celým reťazcom. Graficky je možné znázorniť funkcionalitu programu nasledovne:



Táto schéma zjednodušene zobrazuje spracovanie textu programom. Text je rozdelený na znaky a každý znak je porovnávaný s množinou podporovaných znakov. Ak sa znak nachádza v podporovanej skupine znakov, zvýši sa jeho početnosť v slovníku. Na začiatku je početnosť podporovaných znakov v slovníku nastavená na nulu. Relatívnu početnosť pre jednotlivé znaky program rieši na konci ako podiel absolútnej početnosti znaku a celkového počtu znakov v slovníku. Výsledok je prepočítavaný a zobrazovaný v percentách.

$$\text{RelativnaPoc} = \frac{\text{AbsolutnaPoc}}{\text{ZnakSpolu}} * 100$$

Problematiku spracovania vstupu a prípravu výstupu je možné ukázať na jednoduchom príklade.

Nech sú podporované iba znaky: ***M, m, a*** – zvyšné znaky sa ignorujú.

1. Používateľ zadá text: „***Mama!?***“
2. Program prevedie text na znaky nasledovne: ***'M'***, ***'a'***, ***'m'***, ***'a'***, ***'!'***, ***'?'***
3. Následne začne postup sekvenčného triedenia.
4. Zoberie znak s indexom 0, v našom prípade písmeno ***M***.
5. Skontroluje či je v zozname podporovaných znakov.
6. Znak sa v zozname nachádza, v slovníku sa mu zvýši hodnota o 1.
7. {'M':1}
8. Pokračuje ďalším znakom ***a***.
9. {'M':1, 'a':1}
10. Týmto spôsobom sa analyzuje každý znak zadaného reťazca.
11. Pripravený je výstupný set: {'M':1, 'a':2, 'm':1}
12. Znaky ***!*** a ***?*** sa v tomto príklade ignorujú, nakoľko sa nenachádzajú medzi podporovanými znakmi.
13. Na záver sa vypočíta relatívna početnosť jedinečných znakov.
14. Vypíše výsledok.

Diskusia k výberu algoritmu

Zvolený algoritmus je nenáročný na pochopenie. Vo zvolenom algoritme je jednoduché odsledovať celý proces spracovania dát a poskytuje výrazne väčšiu kontrolu nad výberom znakov. Alternatívami boli využitie samostatných funkcií alebo riešenie problematiky cez štandardné knižnice v jazyku Python, konkrétne použitie knižnice „Counter“. Riešenie by bolo na jednej strane oveľa rýchlejšie no na druhej strane menej transparentné a didakticky neprínosné. Z pohľadu zadania by nespĺňalo základnú podmienku spracovania programu pomocou OOP.

Program

Výsledný kód používa triedu „Analyzator“, ktorá zastrešuje potrebnú funkcionalitu využitú pre analýzu textu. Hlavnou dátovou štruktúrou je slovník, v ktorom sú uložené absolútne početnosti jednotlivých znakov. Množina podporovaných znakov slúži k efektívnej kontrole vstupného textu. Štruktúra triedy je veľmi jednoduchá a jej rozsah zodpovedá požadovanej funkcionalite. Hlavný program na vstupe požaduje iba textový vstup od používateľa. Následne zavolá funkciu na výpočet početnosti a na záver metódu pre výpis výsledkov. Ukážka časti triedy Analyzator:

```python
# Výpočet četností znaků v textu.
# Daniel Kalakay, 2. ročník, B-SGG
# Zimný semester 2025/6
# Úvod do programování  MZ370P19

class Analyzator:
    def __init__(self, text):
        self.text = text #priradenie vstupu do inštančnej premennej pomocou self
        self.pocetnost = {}  # vytvorenie prazdneho slovnika na pocitanie znakov
        self.support = set(
            "AÁÄBCČDĎEÉFGHIÍJKLĽĹMNŇOÓÔPQRŔSŠTŤUÚVWXYÝZŽ"
            "aáäbcčdďeéfghiíjklľĺmnňoóôpqrŕsštťuúvwxyýzž"
            "0123456789"
            ",.?!;"
        ) # nastavenie mnoziny podporovanych znakov
        self.celk_poc = 0   #priradenie celkovej pocetnosti hodnotu 0

    def hlavnafunkcia(self):
        self.pocetnost.clear()  #vycistenie slovniku
        self.celk_poc = 0   #nastavenie celkovej pocetnosti na 0 pred novym pocitanim

        for znak in self.text:
            if znak in self.support:    #kontrola ci je znak podporovany
                self.pocetnost[znak] = self.pocetnost.get(znak, 0) + 1 #ak znak je v slovniku zvys pocet, inak nastav na 1
                self.celk_poc += 1

    def vypis(self):
        zoradene = sorted(
            self.pocetnost.items(),
            key=lambda x: x[1],
            reverse=True
        ) #zoradenie slovnika podla hodnoty absolutnej pocetnosti, zostupne

        print("Znak / Absolútna / Relatívna")

        for znak, abs_cetnost in zoradene: #pre kazdy znak a jeho pocetnost
            relativna = abs_cetnost / self.celk_poc #vypocet relativnej pocetnosti
            relativna=relativna*100
            print(f"{znak} / {abs_cetnost} / {relativna:.1f}% textu")
txt = input("Zadaj svoj text: ") #ui
analyzator = Analyzator(txt)
analyzator.hlavnafunkcia() #spustenie hlavneho programu
analyzator.vypis()  #výpis výsledku
```

Alternatívne prístupy

Autorom uprednostňovanou metódou pre riešenie zadaného problému bolo klasické procedurálne riešenie bez použitia tried, v ktorom by program používal iba funkcie a globálne premenné. Uvedené riešenie by bolo menej prehľadné a ťažko rozšíriteľné, ale za to rýchlejšie na naprogramovanie úspornejšie z pohľadu času a prostriedkov. Ďalším spôsobom by mohlo byť vytvorenie separátnych tried, napríklad vstup, výpočet a výstup. Vzhľadom na (ne)náročnosť zadania by táto voľba bola zbytočne komplikovaná a zdĺhavá.

Vstupné dáta

Vstupné dáta používateľ zadáva ako súvislý textový reťazec (text vo forme slova, vety alebo odstavca). Na dĺžke zadávaného textu nezáleží a použitie rôznych znakov v texte nie je obmedzené. Používateľ nemusí text špeciálne upravovať. Program ignoruje všetky znaky, ktoré nie sú podporované automaticky.

Príklad zadaného textu (najdlhšie slovenské slovo pozostávajúce z 31 písmen): najneobhospodarovávateľnejšími

Výstupné dáta

Výstup programu je vypisovaný do konzoly vo forme riadkov obsahujúcich znak, absolútnu početnosť a relatívnu početnosť v percentách. Relatívna početnosť ukazuje, aký podiel z celkového počtu podporovaných znakov v texte daný znak tvorí. Výstup je zoradený zostupne, najčastejšie znaky sú uvedené ako prvé.

Výstup z príkladu (najneobhospodarovávateľnejšími):

o / 4 / 13.33% textu
a / 3 / 10.00% textu
e / 3 / 10.00% textu
n / 3 / 10.00% textu
j / 2 / 6.67% textu
v / 2 / 6.67% textu
š / 1 / 3.33% textu
b / 1 / 3.33% textu
d / 1 / 3.33% textu
h / 1 / 3.33% textu
i / 1 / 3.33% textu
m / 1 / 3.33% textu
p / 1 / 3.33% textu
r / 1 / 3.33% textu
s / 1 / 3.33% textu
t / 1 / 3.33% textu
á / 1 / 3.33% textu
í / 1 / 3.33% textu
ľ / 1 / 3.33% textu

Priebeh práce

Práca na programe začala na papieri nakreslením schémy návrhu spracovania vstupu a prípravy výstupu. Na jej základe sa riadil proces programovania a spracovania výsledného výstupu ako programu. Počas procesu autor vybral vhodnú štruktúru a následne implementoval hlavnú funkciu pre výpočtu početností a spracovania výstupu.

Po dokončení bol program testovaný na rôznych sadách vstupných textov a boli dokončené úpravy v kóde a formátu výstupu.

Čo by bolo vhodné pridať

Analýzou funkcionalít programu už v procese návrhu sa autor zamyslel aj nad nasledujúcimi otázkami:

Ako by mohol používateľ ovplyvniť výber podporovaných znakov?

Mohol by používateľ nahrať textový súbor priamo z disku?

Akým iným spôsobom by bolo možné pripraviť výstup (napr. CSV)?

Odpovede na vyššie uvedené otázky by mohli byť súčasťou následného rozšírenia programu o nové triedy a metódy, ktoré by umožnili výber vlastného textového súboru používateľom, výber vlastných podporovaných znakov. Ukladanie výstupných dát do externého súboru by prispelo k rozšíreniu možností programu a príprave výstupu pre ďalšie použitie.

Záverečný povzdych

Práca na tomto dokumente autora maximálne vyčerpala, pár krát si počas vytvárania programu pomocou OOP kládol otázku či sa nejedná o snahu používať naberačku na polievku ku konzumácii malého jogurtu. Na druhú stranu ale môžeme konštatovať že výsledný kód vyzerá aspoň kultivovane. Tvorba tohto dokumentu zabrala asi až priveľa času na to aký je program triviálny. Celkovo by autor tento zážitok označil za viac frustrujúci ako uspokojivý, poučný určite ale je. Autor sa naučil, alebo v to aspoň dúfa, písať dokumentáciu k programom a vie že už by to znovu nerobil. Musí uznať že forma vyjadrovania doktora Kryla na neho urobila dobrý dojem.