Univerzita Karlova

Přírodovědecká fakulta

Studijní obor: Geografie a kartografie

Veronika Kalová

Úloha č. 2 – Výpis počtu samohlásek a souhlásek v textu

Úkoly ke zkoušce – Úvod do programování

Kralupy nad Vltavou, 2026

Rozbor problému

„Pro vstupní text zahrnující písmena “A-Ž”, “a-ž”, číslovky “0-9”, speciální znaky “,.?!;” oddělené mezerami, určete počet samohlásek a souhlásek. Spočtěte, jaké procento textu tvoří samohlásky a jaké souhlásky. Výslednou statistiku vytiskněte. Pokud budou v textu nepodporované znaky, ignorujte je.“

V úloze č. 2 bylo úkolem vytvořit program, který by analyzoval text obsahující velká a malá písmena české abecedy i s diakritikou („A-Ž“ a „a-ž“), číslovky („0-9“), speciální „povolené“ znaky (“,.?!;”), mezery a ostatní znaky (v původním zadání popsané jako nepodporované znaky, např. závorka, plus, pomlčka, dvojtečka, procento atd.).

Dále má program umět vypočítat procentuální zastoupení samohlásek a souhlásek v textu, přičemž celkový počet znaků je započítán z písmen, číslic, daných specifických znaků a mezer – nepodporované znaky mají být ignorovány. Tato statistika má být programem vytisknuta. Zadání neobsahuje žádná specifika či omezení ohledně formátu vstupních a výstupních dat.

Existující algoritmy

Existujícím algoritmem, který nebyl pro tvorbu programu zakázán, byla funkce „*.lower()*“ (W3Schools 2026). Tato funkce přepisuje všechna písmena na malá a slouží pro jednodušší práci s textem.

Mezi další existující algoritmy, které by mohly být použity v této úloze, jsou funkce „*.isdigit()*“ a „*.isspace()*“ (W3Schools 2026). Zatímco první ověřuje, zdali znak je číslicí, druhý zjišťuje, jestli je znak v textu mezerou. Využití těchto vestavěných funkcí nebylo nutné pro nízký počet číslic a lehkou identifikaci mezery, funkce by však mohly být použity u složitějších programů nebo pro větší uhlazenost kódu (po estetické stránce).

Popis zvoleného algoritmu

Vytvořený algoritmus pracuje na základě dvou cyklů rozpoznávající znaky v textovém souboru: 1) zdali se jedná o písmeno nebo jiný znak (v tomto případě číslo, mezera nebo speciální znak, pomocí funkce *.isalpha()* (W3Schools 2026), 2) rozpoznání samohlásky a souhlásky. Následně se na základě zařazení znaku do dané skupiny navýší počet daného typu

znaku o 1. Během tohoto cyklu probíhá i převod na textu na malá písmena. V pseudokódu níže je popsán pouze základní algoritmus pro třídění znaků:

```
Pro řádek v dokumentu:
    Pro znak v řádku:
        Pokud je číslo:
            Započti číslo + 1
        Nebo pokud je povolený speciální znak:
            Započti speciální znak + 1
        Nebo pokud je mezera:
            Započti mezera + 1
        Jinak:
            Pokud znak je písmeno:
                Pokud je samohláska:
                    Započti samohláska + 1
                Jinak:
                    Započti souhláska + 1
```

Struktura programu

Program je napsán v objektově orientovaném prostředí. Třída objektu se jmenuje *AnalyzaTextu* a je strukturována do čtyř metod: *otevri_soubor*, *prepocet*, *tisk_vysledku* a *vystupni_soubor*.

Metoda *otevri_soubor* otevírá textový soubor a pomocí algoritmu popsaném v kapitole Popis zvoleného algoritmu rozlišoval jednotlivé znaky a připočítával jejich hodnotu, viz. kapitola Popis zvoleného algoritmu. Během čtení byl použit encoding UTF-8, který podporuje českou abecedu i s diakritikou. Úplně na konci se všechny znaky připisují do listu pomocí *self.novy_soubor.append(znak)*, ze kterého bude následně text přepsán do výstupního dokumentu. Do listu jsou přepsány i „nepodporované“ znaky, které mají být ignorovány. Jelikož nebylo výslovně určeno, jak mají být znaky ignorovány, jsou vynechány pouze z

výpočtu. V *otevri_soubor* nechybí ani blok kódu ošetřující chyby (např. *FileNotFoundError* a *Exception*). Následně *otevri_soubor* volá metodu *prepocet*. *Prepocet* má za úkol přepočítat počet samohlásek a souhlásek v samotném textu na procentuální zastoupení. To je vypočítáno pomocí vzorce:

(počet prvků daného jevu / celková délka textu) * 100

Poté metoda *tisk_vysledku* vytiskne uživateli výsledek ve formátu: „Textový soubor obsahuje X samohlásek (Y %) a Z souhlásek (W %).“ do konzole. Následuje volání metody *vystupni_soubor*, která vytvoří textový soubor s výstupními daty. Poslední dvě metody by mohly být spojeny do jedné metody (pojmenované např. vystup), já je však pro větší přehlednost rozdělila. I přesto, že to nebylo zadáním přímo vyžadováno, výsledky statistiky jsem vytiskla do nově vytvořeného souboru pro případnou další práci.

Popis vstupních a výstupních dat

Vstupními daty pro tento úkol je textový soubor pojmenovaný „*uloha_2.txt*“. Tento soubor obsahuje text obsahující velká i malá písmena, číslice, mezery a výběr povolených speciálních znaků (,.?!;), tak, jak bylo specifikováno v kapitole Rozbor problému. V dokumentu nezáleží na obsahu textového dokumentu (výjimkou je prázdný soubor, tento problém by měl být ošetřen v metodě tisk_vysledku), text může být například rozdělen do více řádků.

Výstupními daty je přímo v programu vytvořený dokument pojmenovaný „*vysledky2.txt*“. V něm je přepsaný text ze souboru „*uloha_2.txt*“ a řádek informující o statistice vyskytujících se hlásek a samohlásek ve formátu: „Textový soubor obsahuje X samohlásek (Y %) a Z souhlásek (W %).“ Hodnoty Y a W jsou v programu pro větší přehlednost zaokrouhleny na první dvě desetinná místa.

Problematická místa

Prvním kritickým místem bylo nejednoznačné a až moc stručné zadání. Ze zadání není zřejmé, z čeho všeho je nutné počítat délku textu (pro výpočet procentuálního zastoupení samohlásek a souhlásek). Běžný uživatel sice přímo kód programu nevidí, ale mohlo by pro něj být sporné, co všechno je počítáno jako text. I například ve Wordu nebo v Google Documents nalezne uživatel dvě různá čísla: jedno číslo vyjadřující délku textu (počet znaků) s mezerami a druhé vyjadřující délku textu bez mezer.

Z důvodu této nejednoznačnosti jsem procentuální zastoupení samohlásek a souhlásek počítala s délkou textu obsahující pouze písmena, nakonec jsem program přepracovala, aby byla statistika počítána z písmen, číslic, speciálních „povolených“ znaků a mezer. Tento problém a jeho možná řešení jsou více rozvedena v kapitole Možná vylepšení.

Mezi dalšími problematickými místy bylo vymyšlení algoritmu pro samotné třídění znaků do skupin „samohláska“, „souhláska“, „číslo“, „speciální znak“, „mezera“ a zbylé „nepovolené“ znaky (nebo také znaky, které je nutné ignorovat). Samotný problém nebyl úplně ve vymyšlení řádků, nýbrž navržení jejich kombinace tak, aby program správně fungoval a nebylo nutné vypisovat všechny znaky. Jednou z možností by bylo vypsat seznam pro úplně všechny znaky, ať už malé či velké, což by však psaní kódu dělalo velice neefektivním. Pozitivem by ale bylo vynechání funkce *.lower()* a přepsání textu i s původními velkými písmeny. Po použití této funkce a nepřepsání textu úplně by někdo mohl namítat, že dochází ke změně dat, zadání to však nezakazuje.

Možná vylepšení

Samotný program by určitě uvítal vylepšení v podobě úplného přepisu původního souboru. Aktuálně výstupní soubor obsahuje text, kde jsou původní velká písmena převedená na malá, protože toto bylo docíleno využitím funkce *.lower()* pro ulehčení identifikace znaků v textu.

Jako jedno z dalších užitečných vylepšení tohoto programu by bylo přidání statistiky pro počítání speciálních znaků nebo číslic v textu i s jejich procentuálním zastoupením – tento problém by se řešil implementací počitadla jako je již u samohlásek a souhlásek.

Jak bylo již zmíněno v kapitole Problematická místa, dále by program mohl být upraven více uživatelsky přívětivěji pro vymýcení nesouvislostí s původním zadáním – například by sám uživatel mohl určit, pro jaké znaky by chtěl statistiku vypsat, zdali by se délka textu počítala pouze z písmen nebo či by se do celkové délky textu počítaly i mezery nebo „nepodporované“ speciální znaky.

V neposlední řadě mě napadá vylepšení programu do podoby počítání jednotlivých znaků – kolikrát se v daném textu vyskytuje daný znak.

Literatura

W3SCHOOLS (2026): Python String isalpha() Method. https://www.w3schools.com/python/ref_string_isalpha.asp (8.2. 2026)

W3SCHOOLS (2026): Python String isdigit() Method. https://www.w3schools.com/python/ref_string_isdigit.asp (8.2. 2026)

W3SCHOOLS (2026): Python String lower() Method. https://www.w3schools.com/python/ref_string_lower.asp (8.2. 2026)

W3SCHOOLS (2026): Python String isspace() Method. https://www.w3schools.com/python/ref_string_isspace.asp (8.2. 2026)