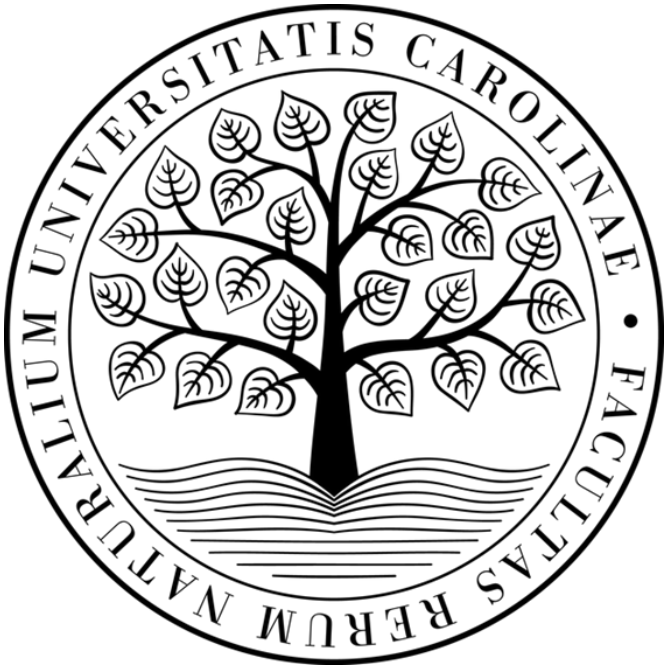
Univerzita Karlova

Přírodovědecká fakulta



ÚVOD DO PROGRAMOVÁNÍ

Zkouška – příklad 2

Počet samohlásek a souhlásek v textu

Martina Pavlová

3 BGEKA

Praha, 2023

Zadání

Ze standardního vstupu načtěte soubor ve formátu TXT obsahující analyzovaný text. Zjistěte množství samohlásek a souhlásek v tomto textu. Vypočtené hodnoty uložte do původního textového souboru.

Popis a rozbor problému

Jednotlivá písmena v textu se dělí na samohlásky a souhlásky. Mezi samohlásky řadíme *a, e, i, o, u, y* i s jejich variantami s háčky a čárkami. Zbylá písmena abecedy považujeme za souhlásky. Pomocí různých programů se dají počítat přímo jednotlivé prvky textu (jednotlivá písmena) nebo právě počet všech samohlásek vyskytujících se v textu či celkový počet souhlásek.

Použitý algoritmus

Každý znak a tím pádem i každé písmeno abecedy má svůj jedinečný ASCII kód. Celá abeceda je zaznamenána v číslech od 65 do 90 pro velká písmena a v intervalu od 97 do 122 pro písmena malá. Problémem u české abecedy je výskyt háčků a čárek nad některými písmeny. Tyto písmena mají své vlastní ASCII kódy nenacházející se v předchozích intervalech. Přehled hodnot ASCII kódů jednotlivých písmen české abecedy s diakritikou lze vidět v tabulce 1.

Program tedy funguje tak, že porovnává jednotlivá písmena textu s uloženými hodnotami samohlásek a souhlásek. Používá k tomu funkci *ord()*, která převádí jednotlivá písmena textu na jejich příslušné ASCII hodnoty (například *ord(a) = 97*). Na základě tohoto porovnání přičte písmeno buďto k celkovému počtu samohlásek či souhlásek anebo jej vynechá.

Samohlásky s diakritikou	ASCII	Souhlásky s diakritikou	ASCII
Á	193	Č	268
á	225	č	269
É	201	Ď	270
é	233	ď	271
Ě	282	Ň	327
ě	283	ň	328
Í	205	Ř	344
í	237	ř	345
Ó	211	Š	352
ó	243	š	353
Ú	218	Ť	356
ú	250	ť	357
Ů	366	Ž	381
ů	367	ž	382
Ý	221		
ý	253		

Tabulka 1: ASCII kódy jednotlivých písmen české abecedy s diakritikou

Struktura programu

Program se skládá z jedné hlavní třídy *Text*, ve které jsou metody ***__init__***, ***count_letters***, ***load***, ***save*** a ***main***:

- ***__init__(self)*** definuje datové položky, se kterými se bude dále pracovat. ***Self.__vowels*** a ***self.consonants*** udávají počáteční hodnotu počtu samohlásek a souhlásek, v tomto případě tedy nula. ***Self.__vowel*** udává ASCII kódy samohlásek a ***self.__vowels_with_diacritics*** udává kódy samohlásek s diakritikou. Poslední ***self.__consonants_with_diacritics*** udává ASCII kódy souhlásek s diakritikou.
- ***Count_letters(self)*** počítá výskyt samohlásek a souhlásek v textu a jejich výsledný počet ukládá do ***self.__vowels*** a ***self.consonants***
- ***Load(self,adress)*** načítá vstupní soubor a data ukládá do ***self.__text***. Zároveň kontroluje, zda jsou data validní.
- ***Save(self,adress)*** ukládá počty samohlásek a souhlásek do původního vstupního souboru
- ***Main()*** jejímž úkolem je volat funkci ***load("input.txt")***, poté ***count_letters()*** a nakonec ***save("input.txt")***

Průběh programu

Program otevře vstupní soubor a načte z něj data. Při jeho otevírání kontroluje, zda jsou data validní, například zda vstupní soubor vůbec existuje či zda k němu má uživatel přístupová práva. Pokud nastane nějaká chyba vypíše se do terminálu chybová hláška a program se ukončí. Pokud je vstupní soubor prázdný, opět se do terminálu vypíše chybová hláška.

Na základě porovnání ASCII kódů jednotlivých písmen textu se seznamy a intervaly samohlásek a souhlásek dojde k přičtení písmene do celkového počtu samohlásek, souhlásek či dojde k jeho přeskočení (pokud nebude v žádném seznam ani intervalu).

Po analyzování celého textu dojde k zapsání celkových počtů do původního vstupního souboru.

Vstupní a výstupní data

Program otevře uživatelem zvolený vstupní soubor ve formátu TXT pojmenovaném *input.txt*. Text musí obsahovat alespoň jedno písmeno latinky, jinak program nemůže nic spočítat.

Výsledný počet samohlásek a souhlásek se po proběhnutí programu zapíše do původního textového souboru, přičemž se zapisuje o dva řádky níže, než je původní text. Dojde tím k na první pohled snadnému rozlišení původního textu od výsledku počítání.

Problematická místa

Největší problém je se znaky, u kterých nechceme, aby je program počítal a aby je tudíž jen přeskakoval. Mezi takové znaky patří například: ! ? „“ - 4. U tohoto programu jsem usoudila, že je méně písmen s diakritikou, která chceme počítat než znaků, které počítat nechceme. Tento problém je zde vyřešen tak, že jsou definované pouze znaky, které chceme, aby program počítal, avšak nejsou to všechny samohlásky a souhlásky, které existují ve všech abecedách a jedná se tedy pouze o znaky vyskytující se v abecedě české.

Tento problém by mohl být řešen i obráceně a to tak, že by byly definované znaky, které nechceme, aby program počítal. Avšak v tomto případě by mohlo hrozit, že na některý ze znaků zapomeneme. V seznamu by se také jistě nenacházely znaky, které neznáme.

Možná zlepšení

Tento program je navržený pro českou abecedu či pro jazyky, které buďto používají pouze základní písmena nebo používají pouze háčky a čárky nad určitými písmeny. Pokud by se v textu tedy nacházely samohlásky jako å, ø nebo ü, program by je mezi samohlásky nezařadil a stejně tak by bylo i v případě některých souhlásek, například ç, ľ, ñ. Tento program by tím pádem mohl být vylepšen tak, aby počítal i se samohláskami a souhláskami jiných jazyků. Buďto by automaticky počítal se všemi písmeny všech abeced anebo by se při spuštění programu vybral jazyk analyzovaného textu.

Dále by se také mohla přidat možnost analyzování více souborů s texty najednou.

Zdroje

Injosoft AB (2023): ASCII table according to ISO-8859-2. https://www.ascii-code.com/ISO-8859-2?fbclid=IwAR1MECArPxvFsFAnTprElcdgVrVo347zNkD5qH6r1PUUzKW2SqhwDZHos1Q (cit. 4. 2. 2023)

Abid, E. B. (2022): What Is the Python ord() Function? How Do You Use It? https://learnpython.com/blog/ord-function-in-python/ (cit. 4. 2. 2023)