

Univerzita Karlova

Přírodovědecká fakulta



Dokumentace k příkladům ke zkoušce

Úvod do programování

Ondřej Prokop

3. ročník

Geografie a Kartografie

Kamenický Šenov, 2026

1. Zadání

Šifrování a dešifrování textu: Caesarova šifra.

Vstupní text zadaný uživatelem zahrnující písmena "A-Z" načtený ze souboru zašifrujte a dešifrujte Caesarovou šifrou se zvoleným posunem. Před šifrováním, ověřte, zda je posun smysluplný, např. abychom se neposunuli o celou abecedu. Ve výsledném textu ponechte mezery a diakritická znaménka a uložte ho do souboru.

Výpočet vzdálenosti bodů $P_1[\varphi_1, \lambda_1]$, $P_2[\varphi_2, \lambda_2]$ na sféře.

Na jednotkové sféře jsou zadány body P_1 , P_2 . Určete vzdálenost obou bodů měřenou po ortodromě procházející těmito body a její azimut. Před výpočtem zkontrolujte vstupní data a neumožněte nekorektní vstup. Zajistěte, aby úloha měla řešení pro oba vstupní body ležící v libovolných kvadrantech.

2. Šifrování a dešifrování textu: Caesarova šifra

Caesarova šifra je algoritmus, který pozmění text tak, že každé písmeno v daném textu je posunuto o stejnou konstantu v abecedě. Nejčastěji je používaný posun o 3, což znamená, že např. z písmene A se po použití šifry stane písmeno D, z B se stane E atd., pokud se používá tradiční anglická abeceda s 26 písmeny.

Úloha je zaměřena na převedení textu do zašifrované podoby, případně na dešifrování textu, pokud předem víme, jaký je posun. Zároveň je v kódu možnost si zvolit, zda má být použita tradiční anglická abeceda s 26 písmeny nebo česká abeceda se všemi písmeny včetně písmen s diakritickými znaménky (dále jako česká abeceda). Pokud jsou v textu například číslice, interpunkční znaménka, mezery nebo odstavce, zůstanou zachovány.

Popis algoritmu

V kódu je definována třída *Sifra*, do které je ukládán text ze vstupního souboru a hodnota posunu, o který bude text změněn. Parametry této třídy jsou však čtyři. Jedná se o *vstup*, *text a posun*, které jsou privátní a *vystup*, který je veřejný. Parametr *vstup* je nezměněný text ze vstupního souboru a parametr *text* je text odpovídající textu ze vstupního souboru, avšak všechna písmena s diakritickým znaménkem jsou převedena na odpovídající písmena bez diakritického znaménka. Tyto dvě třídy byly vytvořeny proto, aby mohlo šifrování a dešifrování probíhat v obou abecedách. Přístup k privátním třídám je řízen podle setterů a @property.

Ve třídě *Sifra* je definováno pět metod objektu:

- *sifrovani_zakl*

- *desifrovani_zakl*
- *sifrovani_cele*
- *desifrovani_cele*
- *export*

První čtyři metody jsou aktivovány podle požadavků uživatele na to, co se má s textem stát. Metody *sifrovani_zakl* a *desifrovani_zakl* využívají tradiční abecedu pro šifrování s 26 písmeny a metody *sifrovani_cele* a *desifrovani_cele* využívají českou abecedu s 41 písmeny. V případě, kdy se jedná o šifrování s anglickou abecedou je k posunu využíváno převedení znaků na ASCII kód. Následně proběhne výpočet a převedení nového kódu zpět na znak.

```
vysledek = (ord(znak) - pocatek + self.posun) % 26 + pocatek
self.vystup += chr(vysledek)
```

Proměnná *pocatek* je ASCII kód znaků „A“ nebo „a“, podle toho, zda původní znak je velké nebo malé písmeno. V druhém případě, kdy se jedná o českou abecedu, je využit řetězec (string) *abc* pro malá písmena nebo řetězec *ABC* pro velká písmena, opět podle původního znaku.

```
nove_poradi = (self.abc.index(znak) - self.posun) % 41
self.vystup += self.abc[nove_poradi]
```

Uživatel má možnost zvolit, zda chce text šifrovat nebo dešifrovat a podle jaké abecedy. Tyto parametry jsou vybírány pomocí proměnné *moznost_sifry*.

Metoda *export* vytvoří nový textový dokument, ve kterém je vypsaný originální text, šifrovaný text a posun, o který byl text pozměněn. Uživatel má možnost zadat název výstupního textového souboru.

Problematické situace

Problém může nastat ve chvíli, kdy zadaný soubor neexistuje. V tomto případě je uživatel vyzván ke vložení jiného souboru .txt. Další problematickou částí je případ, kdy soubor je prázdný, a to je vyřešeno opět výzvou k vložení jiného souboru. Pokud je v textu jiný znak než písmeno, zůstane ve výsledném textu zachován.

Náměty na vylepšení

Kód by mohl být vylepšen například tak, že by sám mohl při dešifrování odhadnout posun a nemusel se tam zadávat.

Dokumentace

[caesarova_sifra.py](#)

Vstupní data

test_text_I.txt – originální text

test_text_II.txt – text k dešifrování podle anglické abecedy a posunem 1

test_text_III.txt – text k dešifrování podle české abecedy a posunem 5

test_text_empty.txt – prázdný textový dokument

Výstupní data

zasif_text_I.txt – šifrovaný originální text podle anglické abecedy

zasif_text_II.txt – šifrovaný originální text podle české abecedy

desif_text_II.txt – dešifrovaný text podle anglické abecedy

desif_text_III.txt – dešifrovaný text podle české abecedy

3. Výpočet vzdálenosti bodů $P_1[\varphi_1, \lambda_1]$, $P_2[\varphi_2, \lambda_2]$ na sféře

Pro výpočet vzdálenosti dvou bodů na sféře se využívá ortodromy. Ortodroma je nejkratší spojnica dvou bodů na sféře. Často se k délce ortodromy uvádí také azimut (úhel, který svírá poledník směrem na sever a ortodroma, ve směru hodinových ručiček).

Pro výpočet délky ortodromy se využívá vzorců:

$$c = \arccos [\cos(90^\circ - \varphi_A) * \cos(90^\circ - \varphi_B) + \sin(90^\circ - \varphi_A) * \sin(90^\circ - \varphi_B) * \cos(\Delta\lambda)]$$

$$d = \pi * \frac{r}{180} * c$$

kde c je středový úhel který svírají polopřímky \overrightarrow{SA} a \overrightarrow{SB} , φ je zeměpisná šířka (-90° - 90°), λ je zeměpisná délka (-180° - 180°), d je délka ortodromy a r je poloměr koule.

Pro výpočet azimutu se pak využívá vzorců:

$$A = \arccos \left(\frac{\cos(90^\circ - \varphi_B) - \cos(90^\circ - \varphi_A) * \cos(c)}{\sin(90^\circ - \varphi_A) * \sin(c)} \right) \quad B = \arccos \left(\frac{\cos(90^\circ - \varphi_A) - \cos(90^\circ - \varphi_B) * \cos(c)}{\sin(90^\circ - \varphi_B) * \sin(c)} \right)$$

Jeden azimut se následně upravuje podle toho, v jakém leží kvadrantu.

Popis algoritmu

V kódu je definovány dvě třídy. První třída je *Bod* s atributy *v* (zeměpisná šířka) a *u* (zeměpisná délka). Druhá třída je *Ortodroma*, která dědí body z třídy *Bod*. Atributy bodů jsou převedeny na radiány a uloženy do atributů třídy *Ortodroma* jako *s1*, *s2* a *d*, což je rozdíl zeměpisných délek obou bodů. *Ortofroma* obsahuje metody *vzdalenost*, *azimut* a *info*.

Metoda *vzdalenost* počítá délku ortodromy. Protože je zde využívána knihovna *math*, kde použité funkce počítají v radiánech, musely být všechny souřadnice převedeny na radiány a vzorec pro délku (viz výše) je upraven na $d = r * c$.

Metoda *azimut* počítá azimuty ortodromy z obou bodů. Azimuty jsou dále upraveny podle kvadrantů, ve kterém se body nacházejí.

Metoda *info* vypíše v *terminal* výsledky všech výpočtů.

V kódu je dále funkce *souradnice*, ve které se dají nastavit hraniční hodnoty zadávaného vstupu a zároveň funkce sama pohlídá, zda při zadání hodnoty nedošlo k chybě.

Problematické situace

Vzhledem k tomu, že se vstup zadává do příkazového řádku, jsou všechny neplatné znaky ošetřené tak, že je uživatel vyzván k jejich opravení. Další problém nastal při testování, kdy při výpočtech vlivem zaokrouhlování vyšel argument pro *acos()* větší než 1 nebo menší než -1. To bylo vyřešeno ohrazením argumentu do intervalu (-1;1) Poslední problém mohl být zadáním dvou identických bodů, to je upraveno jako *Exception* a program se ukončí.

Náměty na vylepšení

Vylepšení by mohlo být například vkládáním obou souřadnic bodu najednou, a ne zvlášť zeměpisné délky a šířky, případně vložením textového souboru, který by nesl souřadnice bodů, případně jejich jména. Další vylepšení by mohlo být zadáním poloměru koule, pokud by uživatel chtěl například zjišťovat ortodromu na jiném tělese, než je planeta Země.

Dokumentace

Vzdalenost_bodu.py

Vstupní data

doporucene_body.txt – zde jsou vypsány souřadnice některých bodů na Zemi v různých kvadrantech, aby šla ověřit funkčnost metody *azimut*

Souřadnice se zadávají jako desetinná čísla, která se v jazyce python píšou s desetinnou tečkou. Zda se jedná o S nebo J šířku či V a Z délku je ošetřeno znaménkem před číslem.

4. Seznam literatury

Homepage of Tomáš Bayer (2026): Teaching. Úvod do programování.

<https://web.natur.cuni.cz/~bayertom/index.php/teaching/uvod-do-programovani> (7.2.2026)

Python Package Index (2026): Unidecode 1.4.0. <https://pypi.org/project/Unidecode/>
(3.2.2026)