# Zkouška z úvodu do programování

1. Zadání
2. Popis a rozbor problému + vzorce
3. Popisy algoritmů formálnímm jazykem
4. Problematické situace a jejich rozbor + ošetření těchto situací v kódu
5. Vstupní data, formát vstupních dat, popis
6. Výstupní data, formát výstupních da, popis
7. Dokumentaci
8. Závěr, možné či neřešené problémy, náměty na vylepšení
9. Seznam literatury.

## Úloha č. 5 (Operace s řetězci)

### Zadání

Nalezení samohlásek v zadaném textu a jejich zvýraznění umístěním do závorek.

Cílem této úlohy bylo ve vstupním textovém souboru vyhledat samohlásky (bez ohledu na velká a malá písmena), uzavřít je do závorek a takto upravený text uložit do nově vytvořeného textového souboru. Rozšířeným zadáním této úlohy bylo rozeznat, jestli je více samohlásek po sobě bezprostředně následujících a v takovém případě je neuzavírat do samostatných závorek ale pouze do jedné společné.

### Popis a rozbor problému, vzorce

Program načte vstupní soubor s textem, kde je potřeba uzávorkovat samohlásky. Nejprve je potřeba rozlišit samohlásky od jiných znaků, k čemuž slouží metoda “isVowel”, která prochází znaky textu a aktuálně načtený znak hledá v zadaném seznamu samohlásek a pokud ho tam najde, vrací hodnotu “True”. Při základní verzi programu by při vracení “True” uzávorkovala daný znak a zapsala ho do výstupního souboru a při vracení “False” zapsala znak do výstupního souboru v nezměněné podobě. Tato verze by ale neřešila uzávorkování více samohlásek jdoucích po sobě, proto je řešení složitější. Řešení tohoto problému je blíže popsáno v bodě č. 4: *Problematické situace a jejich rozbor*.

### Popisy algoritmů formálním jazykem

????

### Problematické situace a jejich rozbor

Nejobtížnějším problémem při tomto úkolu bylo zjistit, jestli se v textu nachází dvě nebo více samohlásek za sebou, které by se měly uzávorkovat společně. Tento problém byl vyřešen pomocí for cyklu. Před samotným for cyklem byla zvolena proměnná “previous\_vowel”, jejíž hodnota se měnila na “True”, pokud poslední načtený znak byla samohláska, jinak nabývala hodnoty “False”.

Poté byly for cyklem procházeny jednotlivé znaky, u kterých bylo pomocí metody třídy “Character” “isVowel” rozhodnuto, jestli je daný znak samohláska a poté zjištěno pomocí proměnné “previous\_vowel”, jestli předchozí znak byla také samohláska. Podle kombinace těchto dvou aspektů byla zvolena jedna za čtyř možných situací, které mohly nastat a podle ní byly k danému znaku umístěny závorky.

### Vstupní data

Text, jenž obsahuje samohlásky k uzávorkování se nachází v souboru pojmenovaném “5\_vstupni\_text.txt”, který je načítán pomocí funkce “open()” a jeho obsah je uložen do proměnné “text”.

### Výstupní data

Výstupní data, tedy text se samohláskami uzavřenými do závorek, jsou uloženy do textového   
souboru “5\_vystupni\_text.txt”.

### Dokumentace

Program nahraje vstupní data v textovém souboru pojmenovaném “5\_vstupni\_text.txt”, který obsahuje text určený ke zpracování. Pokud soubor, který se uživatel snaží otevřít neexistuje nebo je prázdný, program na to uživatele upozorní a skončí.

Program po spuštění vybere samohlásky, které zvýrazní jejich umístěním do závorek a takto upravený text uloží do souboru s názvem “5\_vystupni\_text.txt”.

### Závěr, náměty na vylepšení, neřešené problémy

????

### Seznam literatury

???

## Úloha č. 67 (Počítačová geometrie)

### Zadání

Test, zda bod leží uvnitř konvexního mnohoúhelníku.

Cílem této úlohy bylo načíst z textového souboru souřadnice bodů, které tvoří vrcholy konvexního mnohoúhelníku a poté z téhož souboru načíst zadaný bod a vybranou metodou zjistit jeho polohu vůči mnohoúhelníku – jestli leží vně, uvnitř nebo na jeho hraně a také graficky znázornit danou situaci pomocí vhodného grafického rozhraní.

### Popis a rozbor problému, vzorce

….

### Popisy algoritmů formálním jazykem

Algoritmus užitý v programu je založený na half plane testu.

Metoda half plane test je založená na tom, že hledáme, ve které polorovině zadaný bod leží. Polorovina je dána přímkou a bodem. Pokud tedy chceme zjistit, jestli bod leží uvnitř mnohoúhelníku, můžeme mnohoúhelník považovat za průnik všech polorovin daných hranou mnohoúhelníku a bodem, který leží vždy uvnitř mnohoúhelníku. Takovým bodem je u konvexního mnohoúhelníku těžiště.

Pro výpočet souřadnic těžiště byla použita funkce “najdi\_teziste()”, která načetla seznam souřadnic vrcholů mnohoúhelníku a vypočítala z nich souřadnice těžistě pomocí vzorce:

Souřadnice *x* těžiště = (součet souřadnic *x* vrcholů mnohoúhelníku)/počet hran mnohoúhelníku  
Souřadnice *y* těžiště = (součet souřadnic *y* vrcholů mnohoúhelníku)/počet hran mnohoúhelníku

Poté byly hrany vyjádřeny obecnou rovnicí přímky pomocí funkce “primka\_obecne()”, která vrací koeficienty *a*, *b* a *c*, které jsou součástí obecné rovnice přímky:

Dále byla využita funkce “poloha\_bodu\_k\_hrane()”, která prochází seznam bodů, jimiž jsou určeny hrany a pomocí každé hrany a těžiště mnohoúhelníku vyjádří obecně polorovinu, která je stejná jako obecná rovnice přímky, pouze rovná se je nahrazeno znaménkem nerovnosti v závislosti na konkrétním dosazeném bodě. Do rovnice byl tedy nejprve za *x* a *y* dosazena souřadnicetěžiště a poté souřadnice analyzovaného bodu. Pokud obě nerovnosti byly obě menší než 0 nebo obě větší než 0, znamenalo to, že oba body leží ve stejné polorovině. Pokud nerovnosti platily u všech polorovin daných hranami mnohoúhelníku, pak analyzovaný bod ležel uvnitř daného mnohoúhelníku. Doresit pokud lezi na hrane

### Problematické situace a jejich rozbor

..

### Vstupní data

Vstupní data se načítají ze souboru “67\_vrcholy\_a\_bod.txt”

### Výstupní data

Do terminálu je vypsáno, jestli bod leží uvnitř, vně nebo na hraně mnohoúhelníku

### Dokumentace

…

### Závěr, náměty na vylepšení, neřešené problémy

..

### Seznam literatury

BAYER, Tomáš. *Úvod do výpočetní geometrie. Základní vztahy* [online].   
Dostupné z: <https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk2.pdf>