Technická zpráva úkol č. 1: Point Position Problem

Miroslav Hrubeš, Lucie Peterková

Praha 2023

1 Zadání

Úloha č. 1: Geometrické vyhledávaní bodu
Vstup: Souvislá polygonová mapa n polygonů $\{P_1, ..., P_n\}$*, analyzovaný bod q.*
Výstup: $P_i, q \in P_i$.

Nad polygonovou mapou implementujete Ray Crossing Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího zadaný bod *q*.

Nalezený polygon graficky zvýrazněte vhodnýmm způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textovéhosouboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivýchch polygonů použijteijte špagetový model.

2 Bonusové úlohy

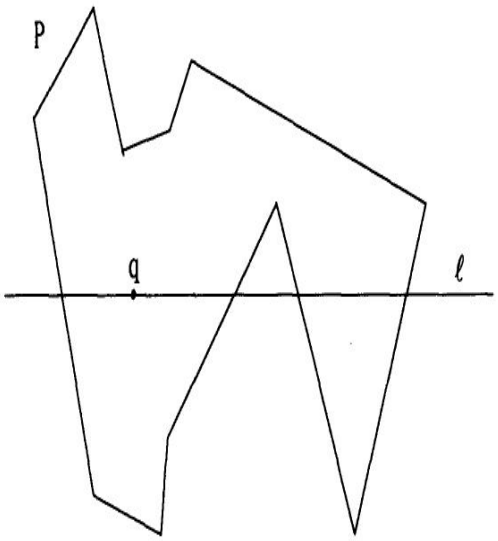
Byly řešeny tyto bonusové úlohy:

- *Analýza polohy bodu (uvnitř/vně) metodou Winding Number Algorithm. (+10b)*
- *Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu. (+5b)*
- *Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu. (+5b)*
- *Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů. (+2b)*
- *Zvýraznění všech polygonů pro oba výše uvedené singulární případy. (+3b)*

3 Point Location Problem

Jedná se o problém hledání polohy bodu uvnitř nebo mimo množinu bodů (polygonu) v rovině. Pro využití v geografických informačních systémech se dá tento problém interpretovat jako zjišťování polohy vůči dalším objektům v prostoru. Z obecného hlediska se jedná o klasický problém v počítačové a výpočetní geometrii.

Huang (1997) popisuje problém následovně: "Máte daný polygon P a libovolný bod q. Určete, zda je bod ohraničený okraji mnohouhelníku P."(obr. 1). I přes zdání jednoduchého zadání problému existuje nemalé množství algoritmů s větší či menší přesností správného výsledku dle konkrétního zadání.



Obrázek 1: Point Location Problem, zdroj: Huang (1997).

Jak již bylo zmíněno, řešení problému je možně pomocí různých algoritmů. Výběr vhodného algoritmu závisí na více faktorech. Jedním z hlavních je typ polygonu, tedy zda se jedné o polygon konvexní nebo konkávní. Polygon je konvexní, když žádný z vnitřních úhlů není větší než 180°. Konkávní polygon alespoň jeden takový musí obsahovat. Dalším faktorem může být možnost ošetření singularit, případně rychlost algoritmů.

3.1 Ray Crossing Algorithm

Metoda se dá přeložit jako Algoritmus průsečíků paprsku. Princip této metody spočívá v tom, že se vede paprsek r procházející zkoumaným bodem q. Následně se zkontroluje, zda se paprsek protíná s hranou polygonu P. Pokud ano, pak se zaznamená počet průsečíků paprsku s hranami polygonu.

$$r(q) : y = yq$$

Na základě počtu průsečíků (k) lze určit, zda se bod nachází uvnitř nebo vně polygonu. Pokud je počet průsečíků lichý, pak se bod nachází uvnitř polygonu. Pokud je počet průsečíků sudý,

pak se bod nachází mimo polygon.

$$k\%2 = \begin{cases} 1, & q \in P \\ 0, & q \notin P \end{cases}$$

Mezi výhody algoritmu Roy crossing patří jeho efektivita při řešení problému point location pro velké datasety polygonů. Algoritmus je schopen vyřešit tento problém v lineárním čase vzhledem k počtu přímek, což je velmi rychlé.

Nicméně, algoritmus Roy crossing má také několik nevýhod, které by měly být zohledněny při jeho použití. Algoritmus je náchylný k chybám při přítomnosti singulárních bodů, jako jsou například body, kde se přímky protínají. Algoritmus také vyžaduje určité úpravy pro práci s polygonem s otvory, což může být pro některé aplikace nevýhodou (Alciatore 1995).

Mezi speciální případy lze zařadit stav, kdy se bod nachází na hraně polygonu či přímo odpovídá vrcholu polygonu. V těchto případech algoritmus jako výstup vrací hodnotu -1.

3.1.1 Pseudokód Roy Crossing

Algorithm 1 *Roy Crossing Algorithm*

Nastav počty průsečíků na 0.
Spočítej délku hran polygonu P.
Pro všechny vrcholy P:
    Sniž souřadnice bodu q tak, aby byl v počátku souřadnic
    Spočítej souřadnice dalšího bodu.
    Pokud aktuální bod odpovídá vrcholu, vrať hodnotu -1.
    Spočítej průsečík úsečky mezi aktuálním bodem a následujícím bodem s osou x.
    Pokud průsečík leží vpravo od bodu q a představuje přechod vpravo přes hranu P, $kr + 1$
    Pokud průsečík leží vlevo od bodu q a představuje přechod vlevo přes hranu P, $kl + 1$.
Pokud bod q leží na hraně P, vrať -1.
Pokud bod q leží uvnitř P, vrať 1.
Jinak vrať 0.

3.2 Winding Number Algorithm

Principem algoritmu je počítání tzv. *winding number* Ω , což by se dalo přeložit jako "číslo ovíjení". Jedná se o to, kolikrát křivka (hrany polygonu P obepíná daný bod q. Pokud se bod nachází uvnitř křivky, bude křivka obepínat bod v určitém směru, Ω bude nenulové . Pokud se bod nachází mimo křivku, nebude křivka bod obepínat a Ω bude nulové (Naresh 2018).

$$\Omega(q, P) = \begin{cases} 0, & q \notin P \\ n > 0, & q \in P \\ n < 0, & q \in P \end{cases}$$

Algoritmus Winding number má několik výhod, které jej činí užitečným pro řešení problému point location. Jednou z největších výhod je relativně jednoduchá implementace algoritmu, což z něj dělá populární volbu pro řešení jednoduchých problémů. Další výhodou algoritmu Winding number je jeho schopnost pracovat s nekonvexními polygony. To umožňuje algoritmu efektivně pracovat s různými typy polygonů.

Mezi nevýhody algoritmu patří jeho neschopnost určit, zda se testovaný bod nachází na hraně polygonu. Další nevýhodou algoritmu může být jeho náchylnost k selhání při přítomnosti polygonů s překrývajícími se hranami nebo s otvory. V těchto případech může být nutné použít jiný algoritmus pro řešení problému. Dalším potenciálním problémem je pomalost algoritmu pro složité nebo velké polygony, kde může být vyhodnocení hodnot Ω časově náročné.

Vzhledem k jeho výhodám a nevýhodám může být algoritmus Winding number vhodnou volbou pro určení umístění bodu v rovině vzhledem k ohraničujícímu polygonu, zejména pro jednoduché a nekonvexní polygony, kde může být použit k rychlému a spolehlivému řešení.

Mezi speciální případy lze zařadit stav, kdy se bod nachází na hraně polygonu či přímo odpovídá vrcholu polygonu. V těchto případech algoritmus jako výstup vrací hodnotu -1.

3.2.1 Pseudokód Winding Number

Algorithm 2 *Winding Number Algorithm*

Nastav hodnotu ϵ na 1^{-19} (velmi malé číslo).
Nastav n na počet vrcholů polygonu P.
$\Omega_s = 0$.
Pro každou dvojici vrcholů P:
    Získej pozici a úhel Ω
    Pokud je poloha bodu q vlevo od přímky, přičti Ω k Ω_s.
    Pokud je poloha bodu q vpravo od přímky, odečti Ω od Ω_s.
    Pokud bod q odpovídá vrcholu, vrať -1,
    Pokud je bod q na přímce a zároveň je rozdíl úhlu a π menší než číslo ϵ, vrať -1 (bod q leží na hraně P).
Pokud je absolutní hodnota rozdílu $|, |\Omega_s| - 2\pi, |$ menší než ϵ, vrať 1 (bod q leží uvnitř P).
V opačném případě vrať 0 (bod q leží mimo$P$).

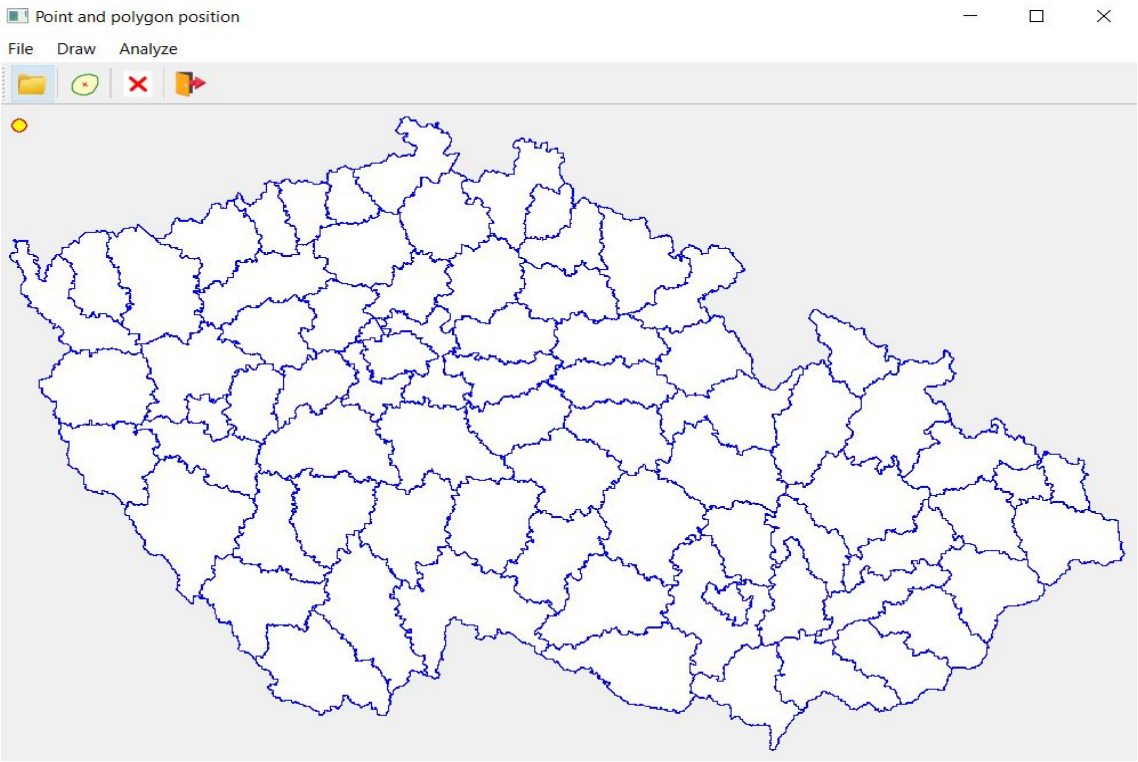
4 Vstupní a výstupní data

Vstupem aplikace je soubor obsahující polygony ve formátu shapefile *(.shp)*. Funkčnost aplikace byla testována s polygonovou vrstvou *okresy.shp*, která zobrazuje okrey Čr.

Aplikace nenabízí výstupní data podobě souboru k uložení. Výsledkem je grafické zobrazení polohy bodu vůči polygonům.

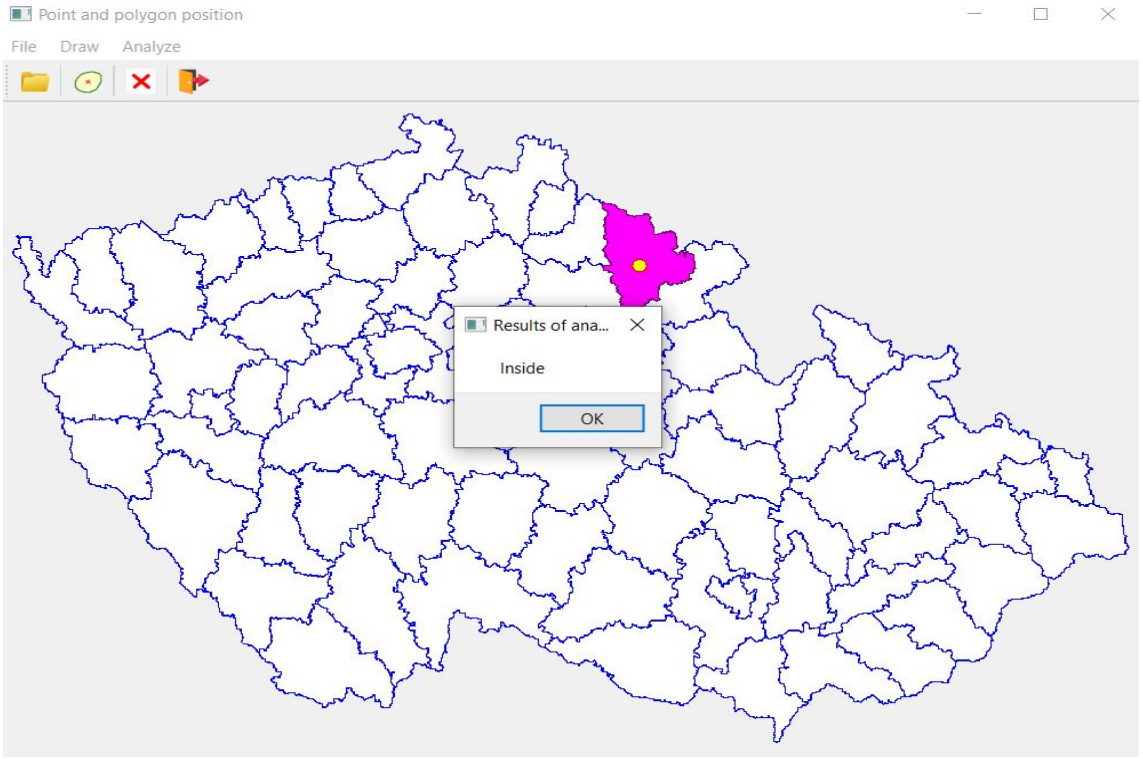
5 Vytvořená aplikace

Obrázek 2 ukazuje grafické rozhraní vytvořené aplikace pro analýzu polohy bodu vůči polygonům. Hlavní okno zobrazuje nahranou polygonovou vrstvu ve tvaru *.shp*. V horní části se nachází tlačítko pro vybraní *.shp* souboru z uložiště počítače. Dále se zde nachází tlačítko pro provedení analýzy polohy bodu vůči polygonu. Tlačítko označené křížkem odstraní nahraný soubor a jako poslední se tu nachází tlačítko exit pro ukončení aplikace. V záložce *Draw* je možné vybrat, zda se bude vykreslovat polygon či bod. Záložka *Analyze* umožňuje vybrat metodu analýzy polohy bodu. Automaticky je nastavena metoda *Winding Number*, ale zde je možné metodu změnit na *Roy crossing*.

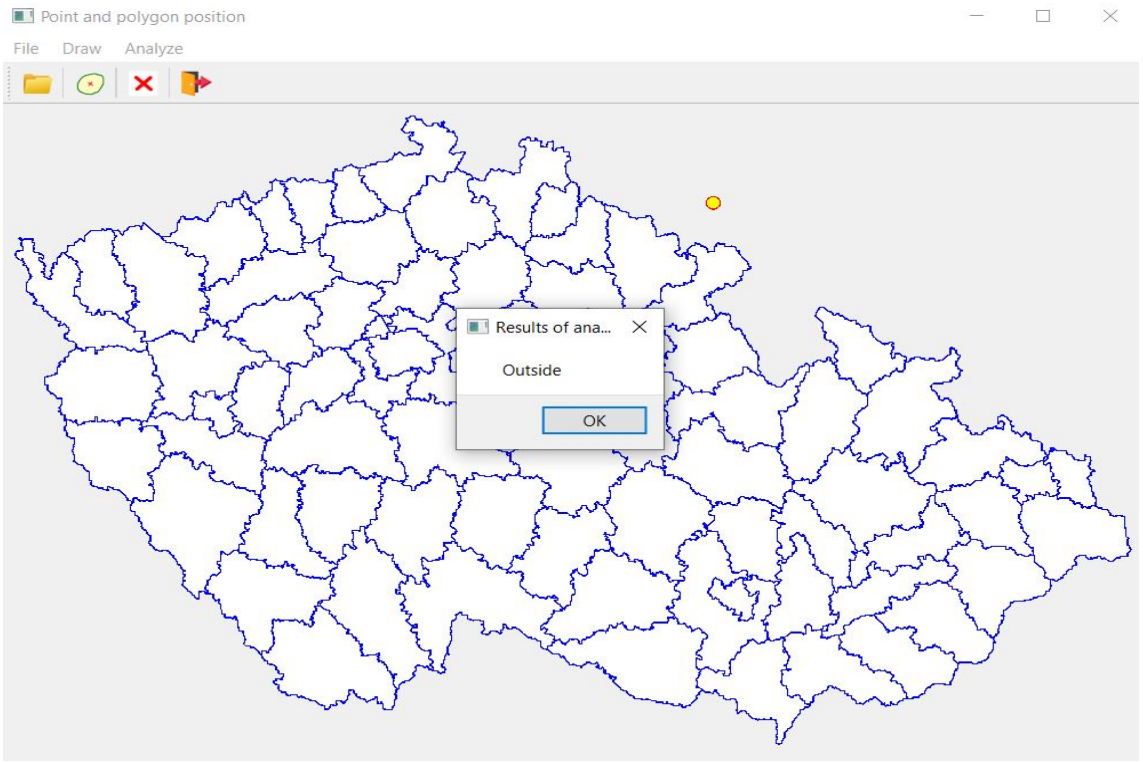


Obrázek 2: Point Location Problem aplikace, zdroj: autoři.

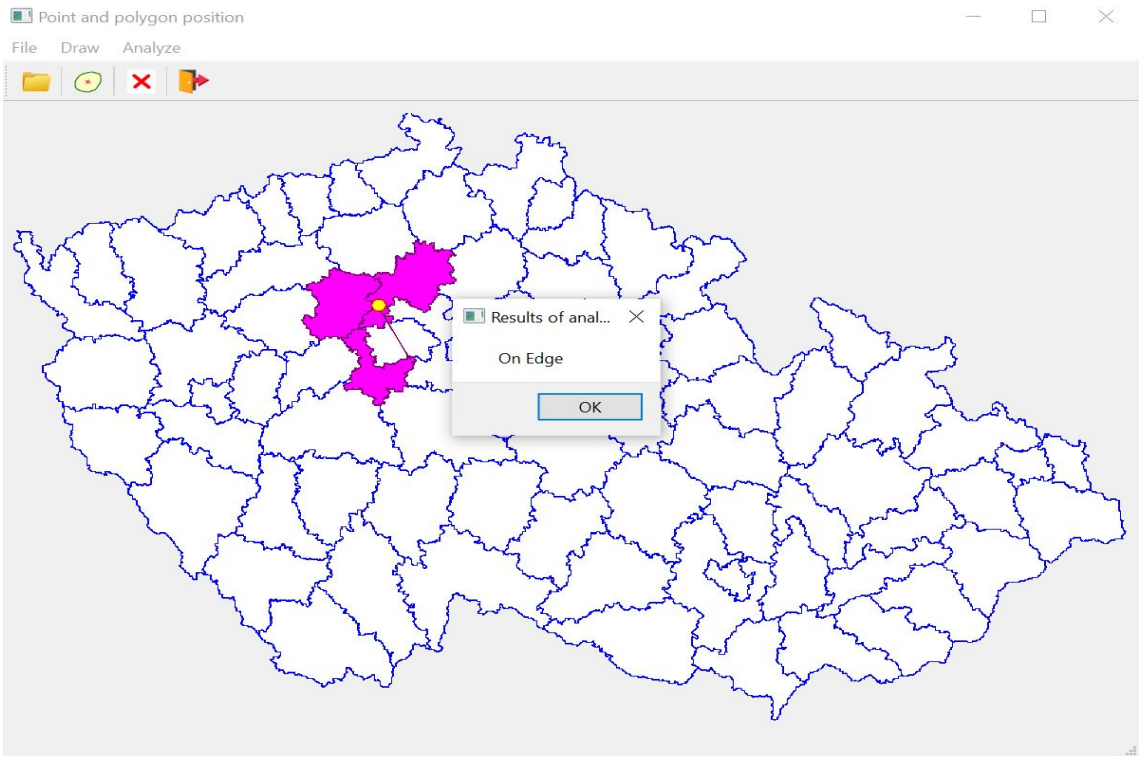
Při provedení analýzy se zobrazí vyskakovací okno s informací o poloze bodu. Na obrázku 3 je zobrazena situace, kdy se bod nachází uvnitř polygonu. Dotyčný polygon je zvýrazněn fialovou barvou. Obrázek 4 představuje situaci, kdy se bod nachází mimo polygon. Obrázek 5 pak představuje stav, kdy se bod nachází na hraně polygonů (resp. odpovídá vrcholu)



Obrázek 3: Bod v polygonu, zdroj: autoři.



Obrázek 4: Bod mimo polygon, zdroj: autoři.



Obrázek 5: Bod na hraně polygonů, zdroj: autoři.

6 Dokumentace

Aplikace je tvořena třemi skripty. Uživatelské rozgraní je tvořeno skriptem *mainform.py*, jehož základ byl vytvořen pomocí softwaru *QT Creator* . Vizualizaci a vykreslování objektů zajišťuje skript *draw.py*. Skritp *algorithms.py* definuje matematické metody pro správné provedení zvolených analýz.

Metody skriptu *mainform.py* :

- **setupUi**: inicializuje a nastavuje různé prvky uživatelského rozhraní, jako např. velikost okna, hlavní widget, menu, lištu nástrojů a akce (např. otevření souboru, ukončení aplikace, vykreslení polygonu, analýza pozice bodů a polygonů apod.).
- **retranslateUi**: nastavení textů pro různé prvky uživatelského rozhraní.

Metody skriptu *draw.py* :

- **setPath**: načte cestu ke souboru .shp pomocí dialogového okna, přečte soubor a vytvoří polygon.
- **mousePressEvent**: spustí se po kliknutí levým tlačítkem myši. Metoda uloží souřadnice bodu, do kterého se kliklo do proměnných x a y, přidá nový bod p do polygonu a překreslí obrazovku.
- **paintEvent**: spustí se při vykreslení okna. Vytvoří grafický objekt, nakreslí polygon a zvýrazní polygon obsahující daný bod.
- **switchsource**: přepíná mezi režimy přidání bodu a přidání vrcholu polygonu.
- **getPoint**: vrací aktuální pozici bodu jako QPointF.
- **getPolygon**: vrací standardní polygon jako seznam QPolygonF.

- **getResPol**: přidá výsledný polygon jako QPolygonF do seznamu.
- **clearPol**: vymaže všechny standardní polygony.
- **clearPoint**: vymaže aktuální bod.
- **clearResPol**: vymaže všechny výsledné polygony.

Metody skriptu *algorithms.py* :

- **getWindingNumber**:vypočítává počet ovinutí (winding number) bodu q vzhledem k polygonu P. Nejprve vypočítá pozici bodu q vůči každému vrcholu polygonu P a poté sčítá úhly, které se vyskytují mezi bodem q a každou hranou polygonu P. Pokud výsledný úhel sečtený z celého polygonu P odpovídá úhlu 2 π, bod q leží uvnitř polygonu P a funkce vrátí hodnotu 1. V opačném případě se bod q nachází mimo polygon P a funkce vrátí hodnotu 0.
- **getPointPolygonPositionR**: určuje polohu bodu q vzhledem k polygonu P. Metoda prochází všechny vrcholy polygonu P a pro každý z nich určuje, na které straně přímky procházející bodem q a tímto vrcholem se bod q nachází. Pokud leží bod q na hraně polygonu P, funkce vrátí hodnotu 1. Pokud leží bod q uvnitř polygonu P, funkce také vrátí hodnotu 1. V opačném případě, když se bod q nachází mimo polygon P, funkce vrátí hodnotu 0.
- **getRayCrossing**: určuje, kolikrát přímka procházející bodem q a bodem s následujícím vrcholem polygonu P protíná hrany polygonu. Pokud počet průsečíků je lichý, bod q leží uvnitř polygonu P, a funkce vrátí hodnotu 1. Pokud počet průsečíků je sudý, bod q leží mimo polygon P, a funkce vrátí hodnotu 0. Pokud bod q leží na hraně polygonu P, funkce také vrátí hodnotu 1.

7 Závěr

Úloha si kladla za cíl vytvořit aplikaci pomocí Qt Creator a kriptů v pythonu, která bude analyzovat polohu bodu vůči nahraným polygonům ve tvaru *.shp* pomocí metod *Roy crossing* a *Winding number*. Aplikace byla doplněna o bonusové úkoly, kteeré řeší singulární případy.

Byla vytvořena přehledná a jednoduchá aplikace pro prevedení námi zvolené analýzy, která dává správné výsledky i co se týče singulárních případů.

Pro další vylepšení aplikace by bylo možné rozšířit odpověď programu na otázku, kde se bod vůči polygonu nachází (*outside* / *inside*) na další varianty, jako například *on boundary* / *on vertex* atd.

8 Zdroje

Přednášky z předmětu *Algoritmy z počítačové kartografie.*

ALCIATORE, D., MIRANDA, R. (1995): A Winding Number and Point-in-Polygon Algorithm. Department of Mechanical Engineering. Colorado State University.

HUANG, C., W., SHIH, T., Y, (1997): On the complexity of point-in-polygon algorithms. Computer & Geoscienses, 23, 1, 109-118.

NARESH, K., MALLIKARJUN, B. (2018): An Extension to Winding Number and Point-in-Polygon Algorithm. IFAC-PapersOnLine. 51, 1.