návod

po spuštění programu je do konzole vypsán úvodní text

pokud je aplikace otevřena pomocí .txt souboru, je soubor považován za vstup.

dále pokud je ve složce s aplikací soubor s názvem vstup.txt, načte program jej.

pokud ani jedno, otevře se dialog pro vybrání souboru

soubor je nutné načítat ve formátu:

první řádek dvě čísla oddělená pouze pomlčkou (první menší nebo rovno druhému),

druhý řádek trasa prvního řidiče, třetí řádek trasa druhého řidiče,

kroky trasy oddělené čárkou, bez přebývajících znaků, bez čárky za posledním (např. "50W,80S,99E")

krok se musí skládat z čísla a písmena W,S,E nebo N, číslo musí být první, bez mezery.

(soubor vstup.txt přiložený k programu splňuje tyto požadavky)

pokud interval přesahuje za dojezd některého z řidičů, jsou uvažovány kroky do konce trasy

algoritmus a myšlenkový postup

procházet trasu kterou řidič projel bod po bodu a srovnávat s trasou druhého řidiče by bylo výpočtově extrémně náročné, proto musím hledat bod po rovných úsecích. (takovéto podobě jsou i data načítána)

trasa je udávána pouze pomocí jednotlivých kroků (řekněme jako vektory ve dvou kolmých směrech), proto abych mohl úseky srovnávat musím převést trasu na absolutní souřadnice.

na začátku ze souboru načtu interval a dva řetězce dráhy.

textový řetězec dráhy rozdělím na úseky podle čárek na string[], deklaruju počáteční bod (0.0)

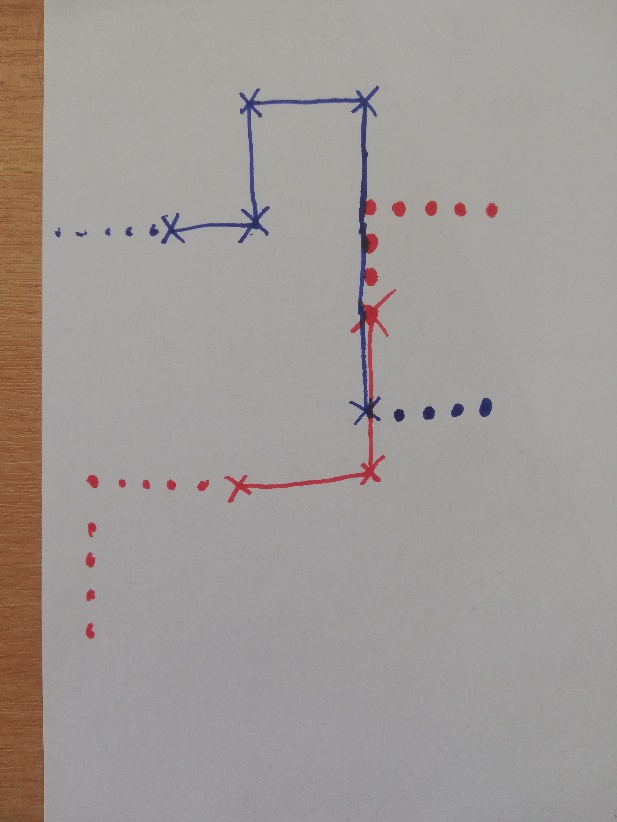
potom procházím každý úsek a přepočítávám polohu, přičítám k ujeté vzdálenosti, úsek ukládám do proměnné, kterou jsem si definoval (nižší souřadnice, vyšší souřadnice a konstantní souřadnice) a přidávám do seznamu (horizontálních a vertikálních úseků).

jelikož hledáme bod pouze v daném intervalu, není třeba uvažovat začátek a konec trasy, ukládám až úseky po překročení začátku intervalu (+ úsek vstupu do intervalu a totéž při výstupu z intervalu) a končím výpočet polohy po vystoupení z intervalu, protože již není co hledat za koncem intervalu.

poté co takto převedu obě trasy porovnávám horizontální úseky z první trasy s vertikálními z druhé trasy a naopak. Ve chvíli, kdy najdu první bod ukončuju hledání.

*při srovnávání dvou kolmých úseků zjišťuji, jestli je vždy konstantní pozice jednoho úseku mezi nižší a vyšší z druhého úseku*

Evidentně pokud by dva rovnoběžné úseky měly společný bod, najdu minimálně jeden další společný bod mezi jedním z nich a následujícím, nebo předchozím z druhého – tedy nemusím je porovnávat (to platí pouze v případě, že navazující úseky nejsou v opačném směru). výjimku z tohoto tvoří konce a začátky ořezaných tras, kde v případě že jsou ve stejném směru a na úseku, kde se prolínají není odbočení. (viz. obrázek 1, tečkovaně trasa mimo interval, křížky značí krajní body)



Obrázek 1

proto pokud předchozí algoritmus nenajde žádný bod, porovná program ještě krajní úseky (obou seznamů/polí, tedy i některé navíc) a až potom vydá výstup o neexistenci bodu.

*při srovnávání rovnoběžných kontroluji, zda jsou shodné konstantní a jestli se překrývají, a pokud ano vyberu krajní bod, který je uvnitř (teoreticky oba jsou, pokud nejsou nalezeny jiné v předešlém algoritmu, ale přesto se překrývají*

**Poznámka k zadání**

V zadání není přímo určeno, že by po sobě jdoucí úseky měly mít na sebe kolmý směr (řidič třeba může jet 10 na sever a potom 5 na jih), ale jelikož v testovacích datech tomu tak je, využívám toho ve svém algoritmu. Pokud bych měl přesně vyhovět zadání, musel bych místo pouze koncových úseků srovnávat všechny rovnoběžné úseky (tedy každý s každým)

I přesto je vhodné rozdělit úseky podle směru, protože díky tomu můžu použít jednodušší logiku porovnávání.

**Jiná řešení**

1. mohl jsem, tak jako je naznačeno v zadání, hledat všechny body na trase a pak až vybrat ty z rozsahu, to by zjednodušilo a zpřehlednilo algoritmus načítání, ale docházelo by k mnoha zbytečným výpočtům.
2. nemusel jsem rozdělovat úseky na vertikální a horizontálním, a potom srovnávat všechny úseky se všema.
3. úseky mohly být uloženy v podobě dvou bodů (počátečního a koncového), to by zobecnilo vyhodnocování pro horizontální a vertikální úseky, ale následný vypočet porovnání by byl složitější

Všechna tato zjednodušení by ubíraly efektivitě algoritmu.