UNIVERZITA PARDUBICE

Datové struktury

Semestrální práce A

Matěj Trakal

24.3.2012

Dynamický výpočet optimální trasy přemístění.

Vypracování semestrální práce

Pro semestrální práci jsem zvolil programovací jazyk C# a IDE Visual Studio 2010.

1 Použité datové struktury

Jelikož byl kladen důraz na optimalizaci a struktur, začal jsem se zabývat jejich zkoumáním a využitím. Postupně jsem se dostal až k tomu, že nejvhodnější strukturou pro ukládání neorientovaného, jednocestného grafu bez záporných hran bude struktura graf.

Pro vyhledávání po vyloučení některých algoritmů (Floyd-Warshall a Bellman-Ford) mi zůstali poslední dva a to Dijkstrův a A*. Z nich jsem nakonec zvolil Dijkstrův algoritmus.

1.1 Graf

Strukturu grafu jsem postavil na dvou slovnících, které mají pro vkládání a odebírání dle klíče složitost O(1) a v nejhorším případě (pokud kapacita slovníku je menší, než počet prvků v něm) složitost O(n).

Slovníky jsou použity pro uchování vrcholů a hran samostatně.

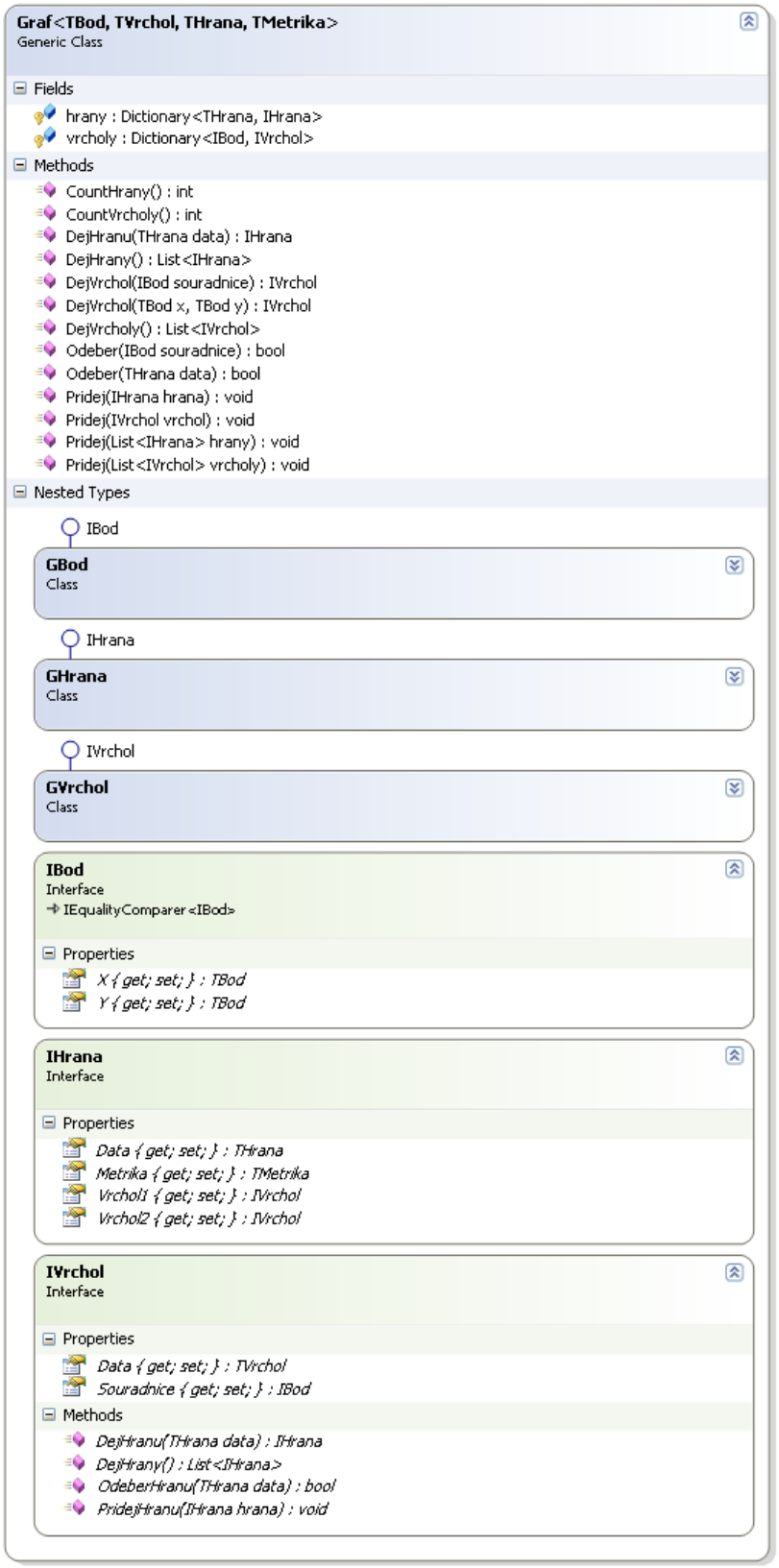
1.1.1 Hrany

Hrana je třída složená z generické vlastnosti Data pro uchování datové složky, dvou generických Vrcholů a jedné generické Metriky, slouží jako datové úložiště a kromě operací poskytovaných vlastnostmi nic nedělá.

1.1.2 Vrcholy

Třída vrcholů obsahuje opět datovou složku, souřadnice odvozené od rozhraní IBod a následně obsahuje seznam Hran se kterými je vrchol spojen.

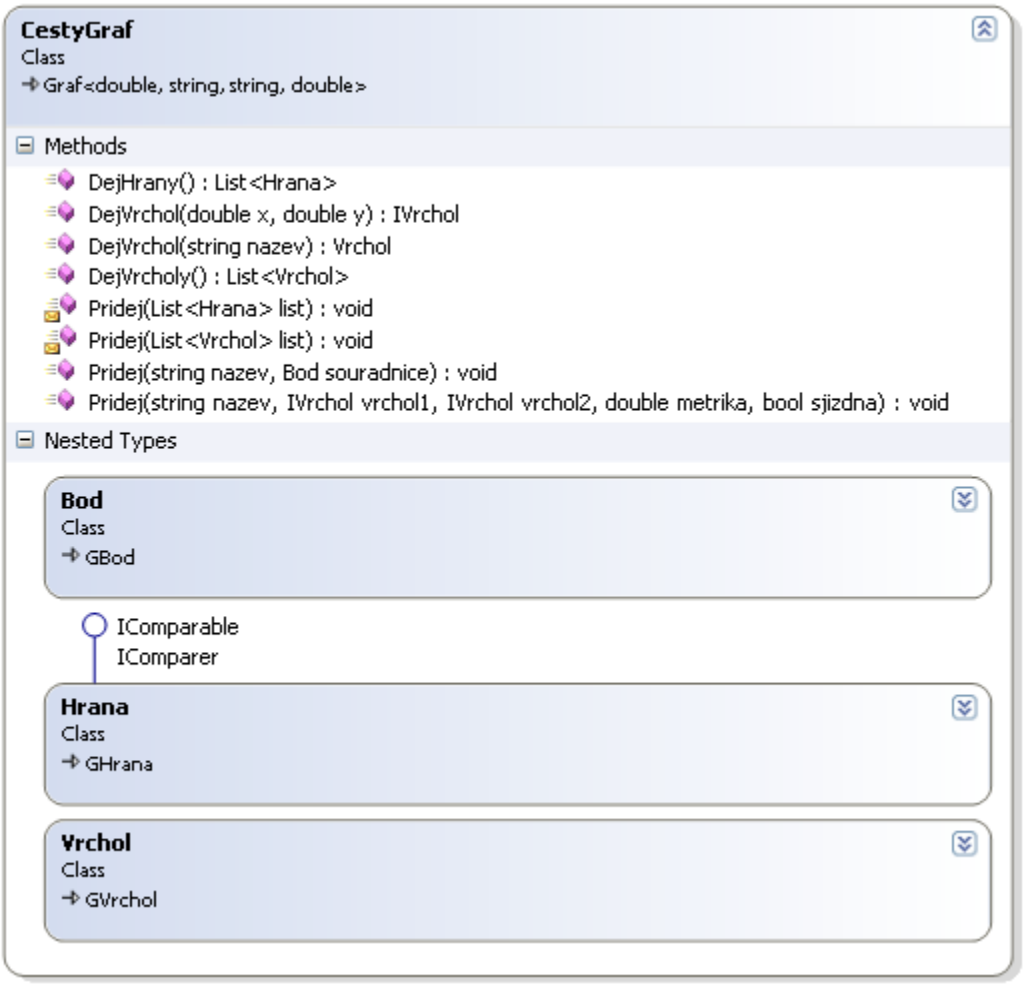
Seznam hran je realizován pomocí ADT List, kde operace přidání má složitost O(1). Většina ostatních operací, kromě přístupu pomocí indexu O(1), složitost O(n). Naštěstí se těchto operací využívá pouze při změně hran, což není klíčová funkce této aplikace, tedy to nevadí.



Obrázek 1: ADT Graf

1.2 CestyGraf

Je odvozená třída od ADT Graf popsaného výše, kde již nepoužívám generické typy, ale specifické, dle potřeb aplikace a přidávám některé další operace, které ADT neobsahuje (konstruktory, porovnávání).



Obrázek 2: CestyGraf odvozené od ADT Graf

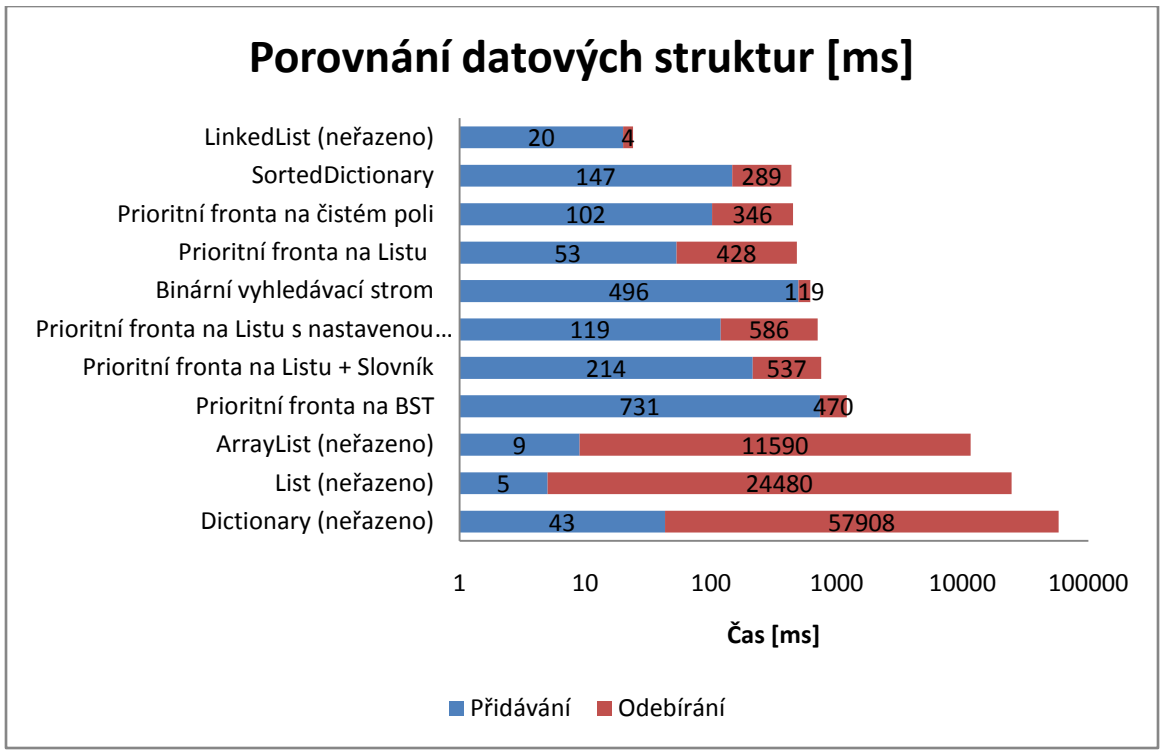
1.3 Prioritní fronta

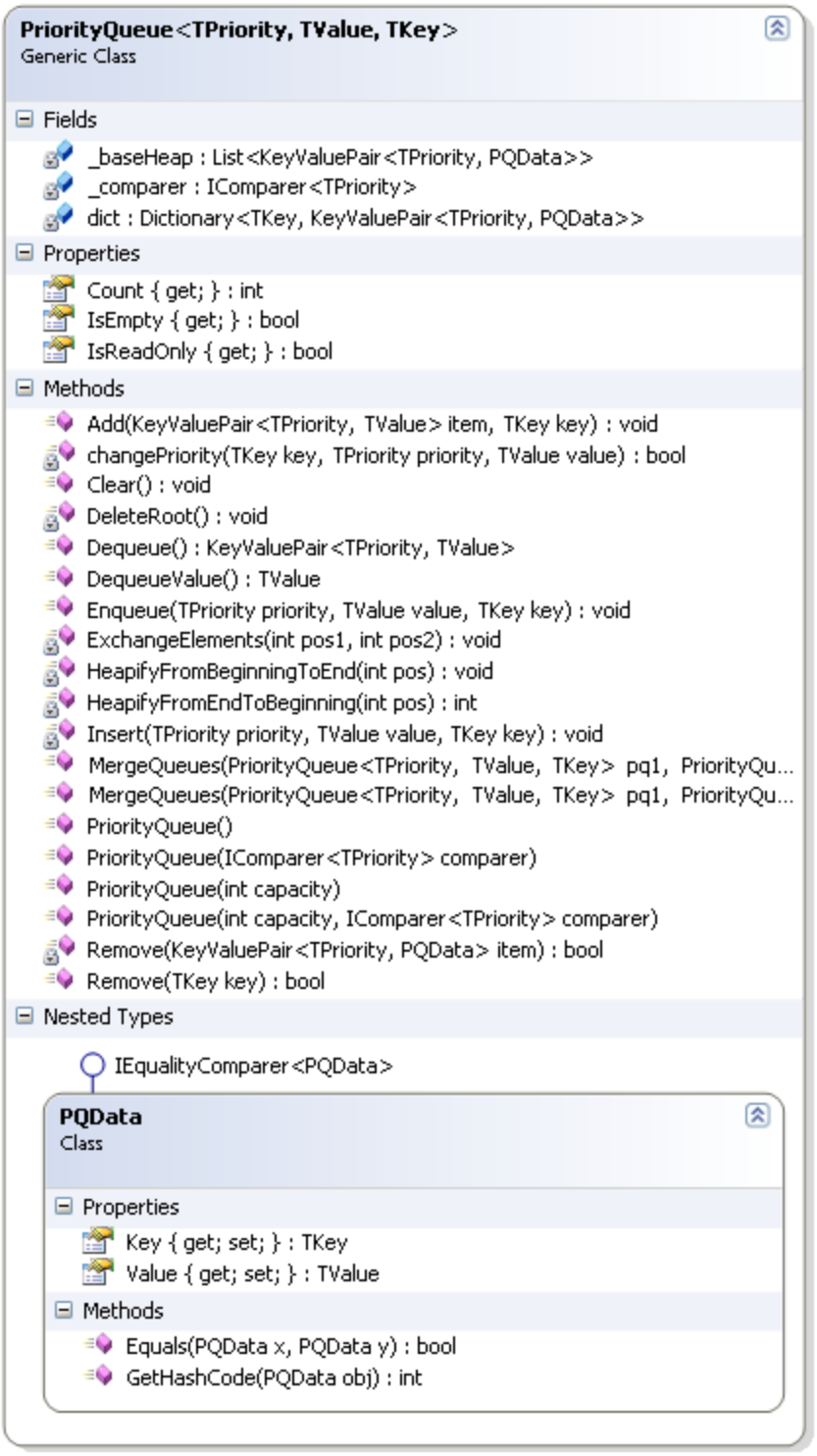
S touto strukturou jsem měl drobné problémy s implementací, nakonec jsem se uchýlil k využití setříděného pole, které mi vycházelo jako nejlepší při porovnání několika podkladových struktur.

Pro otestování jsem si vytvořil projekt, který vkládal do struktur datový typ double jako klíč i jako data a po seřazení je opět vracel a odebíral ze struktury. Celkový počet vkládaných unikátních hodnot 99936.

Graf jsem musel použít s logaritmickým rozložením, jinak data vůči jiným byla zanedbatelná. Z grafu je patrné, že nejrychlejší struktura pro řazení je SortedDictionary, následován prioritní frontou na čistém poli a v závěsu ostatní struktury. Bohužel čisté pole mi nevyhovovalo (neumožňuje rychlý přístup k prvku pomocí klíče (ne priority), proto jsem použil upravenou prioritní frontu se slovníkem, která vychází ještě celkem dobře časově a při použití s nastavenou kapacitou dosahuje ještě lepších výsledků).

Některé struktury jsem neměl trpělivost nechávat seřadit (řazení trvalo přes minutu, proto jsem na řazení dále nečekal).

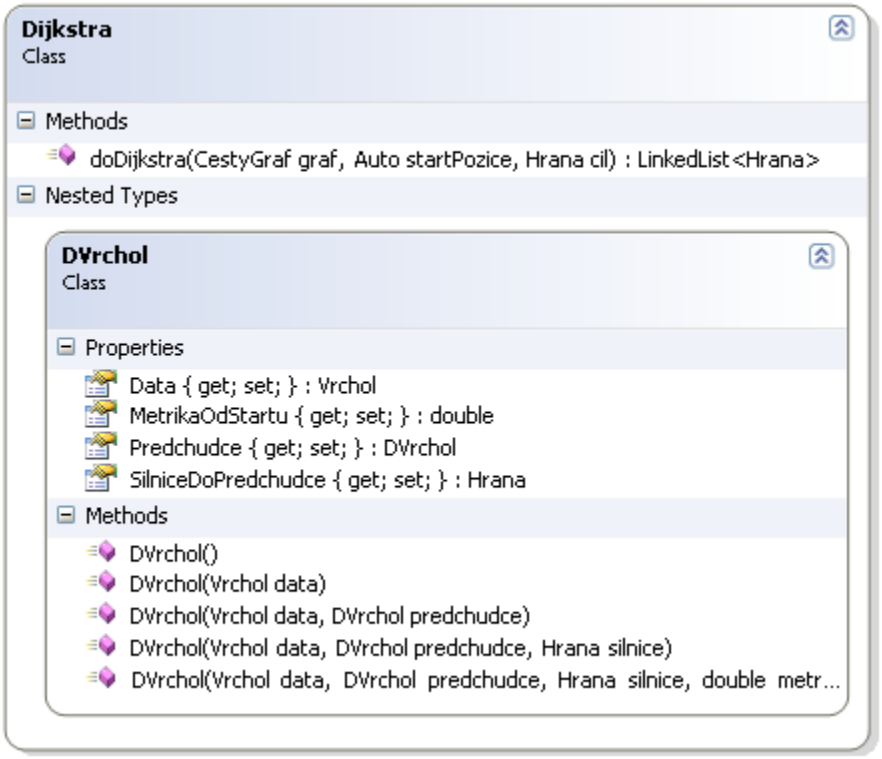




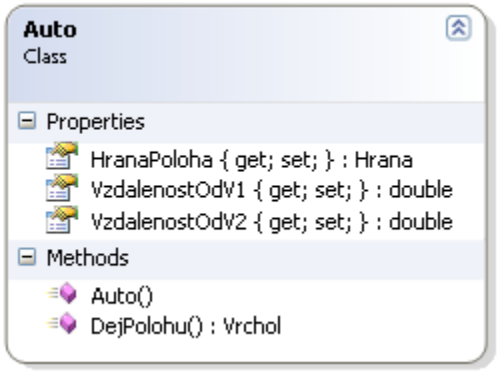
Obrázek 3: Prioritní fronta

2 Vyhledávací algoritmus

Pro vyhledání nejkratší cesty ke koncové hraně využívám upravený Dijkstrův algoritmus. Hlavní úprava spočívá v nezastavení po nalezení první cesty, která by byla sice nejkratší na počet vrcholů, ovšem nemusela by být optimální cestou v rámci ohodnocení pomocí metriky (kilometrů).



Obrázek 4: Třída zapouzdřující operaci s vyhledávacím algoritmem



Obrázek 5: Struktura auta po vložení do mapy

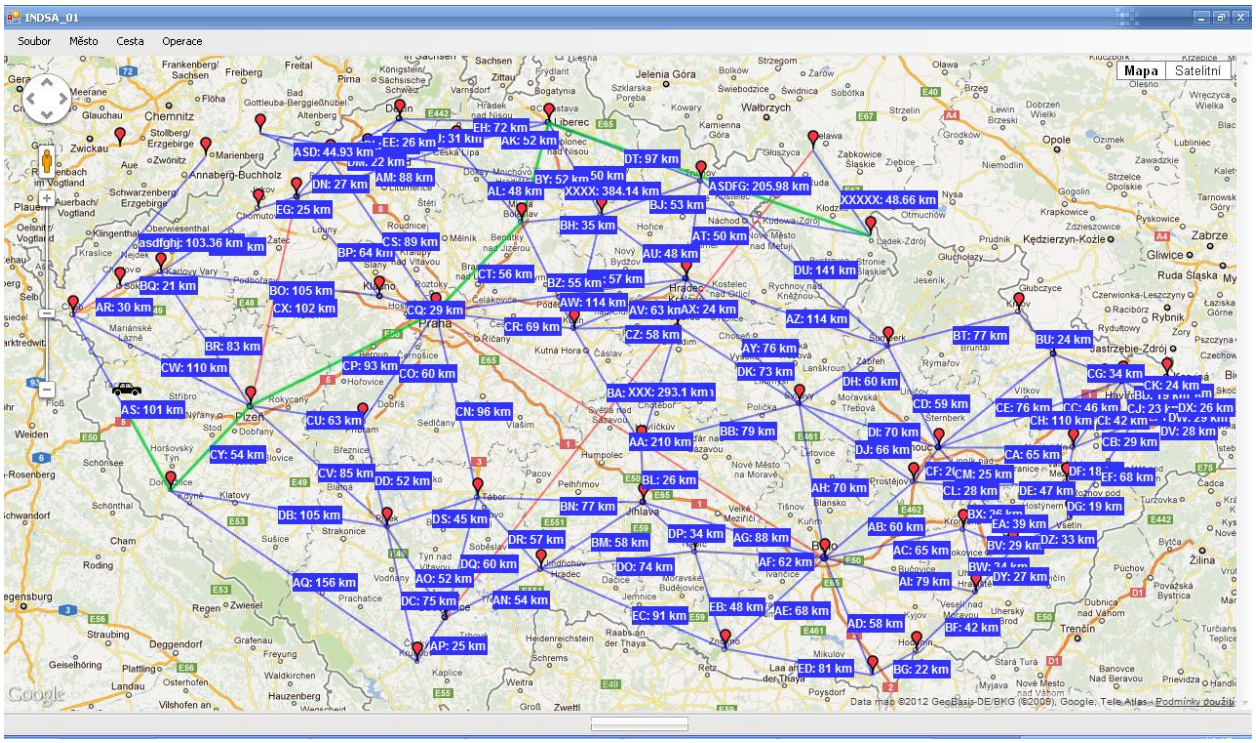
3 GUI

Pro vykreslování cest a vrcholů využívám podkladových map společnosti Google.

Po načtení podkladových souborů zůstává soubor pro ukládání stále otevřen pro zápis (ostatní aplikace mají právo čtení) a mapa je průběžně při změnách ukládána, tedy nemůže dojít ke ztrátě dat při případném ukončení aplikace bez uložení.

V mapě jsou zobrazeny červeně nesjízdné silnice, zeleně nalezená cesta od umístění vozidla do cíle a modře obyčejné cesty, které je možné využít.

Pro získání souřadnic města a jeho názvu stačí kliknout do mapy a zvolit z menu Město > Přidat, čímž se zjistí pozice klinutí do mapy a stáhnou se informace o lokaci.



Obrázek 6: Výsledná aplikace se zobrazenou nalezenou cestou.

Jelikož využívám přímo souřadného systému GPS pro určení polohy, musel jsem se zabývat výpočtem vzdáleností. Pokud bych použil pro výpočet např. obyčejný výpočet pomocí Pythagorase a vzdálenosti dvou bodů v rovině, na geoidu[i] bych postupným vzdalováním od rovníku dostával nepřesné údaje. Proto jsem zvolil výpočet pomocí ortodromy[ii]. Ten jsem nalezl vyřešen na Internetu a upravil pouze pro použití v programovacím jazyce C#.

[i] Geoid je fyzikální model povrchu Země při střední hladině světových oceánů.
[ii] Ortodroma je nejkratší spojnice dvou bodů na kulové ploše.