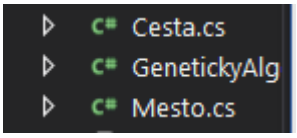
Druhá průběžná práce INUI1

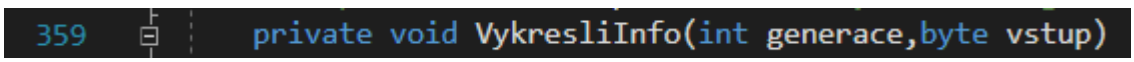
Autor práce: Miroslav Hájek

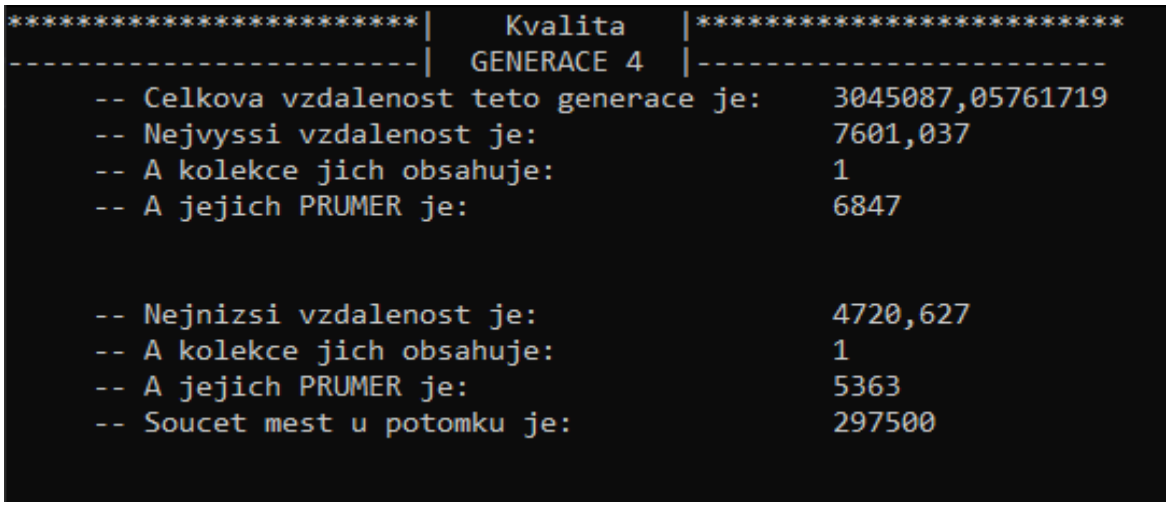
Popis projektu

- Pro řešení semestrální práce jsem se rozhodl využít programovacího jazyku C# a IDE Visual Studio pro vývoj
- Projekt jsem se rozhodl rozdělit do tří tříd, podle úrovně, kterou mají reprezentovat. Nejnižší úrovní je třída Mesto, dále se tyto třídy sdružuji do třídy Cesta (obsahuje 35 prvků (**Mesto**)). S touto třídou je poté pracováno ve třídy GenetickyAlgoritmus.
    - **Mesto** (třída, která reprezentuje jedno město a uchovává na sobě určité informace)
    - **Cesta** (třída, která reprezentuje řetězec 35 měst, představující cestu obchodního cestujícího)
    - **GenetickyAlgoritmus (**hlavní třída programu, která realizuje prvky genetického algoritmu. Využívá třídy Město a Cesta pro svoji funkčnost**)**



- Veškeré třídy, metody a atributy jsou velmi komentované již ve zdrojových kódech, proto tento report je spíše shrnutím, jelikož kódy již obsahují značné množství informací, tak aby byly dostatečně přehledné.
- Hlavní třída **program**.cs obsahuje pouze vytvoření instance **GenetickyAlgoritmus**. Jelikož veškeré metody jsou zakomponovány v konstruktoru této třídy, není nutné volat žádné jiné obslužné metody a program se tak vykoná automaticky.
- **Při spuštění programu je nutné, aby excelová tabulka obsahující vzdálenosti měst, byla umístěna na cestě** `d:\Distance.xlsx.` Při spuštění programu jsou totiž za pomoci metody **DefinujMapuVzdalenosti** spočítány vzdálenosti měst mezi sebou. Pokud by nebyl tento soubor umístěn na danou pozici, nebude program fungovat. Případná úprava cesty tohoto souboru je možná ve **třídě GenetickyAlgoritmus na řádku 457.**
- Program je demonstrován vykreslením do konzole, kde jsou zobrazeny nejdůležitější informace o právě probíhaném kroku. Na nejvyšší řádku je vidět jaká operace právě probíhá, aby bylo možné sledovat jaký vliv mají dané operace na výslednou vzdálenost. Možnostmi jsou Kvalita (ohodnocování kvality potomků, neboli Fitness funkce), NovaGen (vytvoření nové generace pomocí křížení rodičů mezi sebou) a Mutace (jedná se o mutaci jednoho ,,genu,, ve sbírce potomků, tedy je změněno jedno město). Druhý řádek ukazuje o kolikátou generaci se jedná. Třetí řádek označuje celkovou vzdálenost v km celé generace potomků. Čtvrtý řádek označuje nejvyšší vzdálenost měst, které je obsažena v aktuální generaci, dále je počet těchto potomků v generaci a šestý řádek poté označuje průměrnou vzdálenost 20 % potomků s nejvyšší hodnotou. Další čtyři řádky jsou obdobou výše zmíněných, avšak vypovídající o spodní hranici, tedy nejlepší potomci. Veškeré zobrazení do konzole poté vykonává metoda **VykresliInfo.**





- **Spuštění programu**
  - **Windows 10**
    - **Adresář lze stáhnout:**
    - https://drive.google.com/drive/folders/1Ld3dCUTM1gW52THS1UcUOo1Sswk_vqT-?usp=sharing
    - Tento soubor se nachází v adresáři aplikace, konkrétně v Mesta\Mesta\bin\Debug\Mesta.exe
  - **Linux (Ubuntu)**
    - **Využití technologie Mono, která slouží pro multiplatformní aplikaci napsanou v C# (www.mono-project.com)**
    - Poté v termínálu linux zapsat následující příkazy
      - **sudo apt-get install mono-devel**
      - **sudo apt-get install libgdiplus**
    - A samotné spuštění programu pomocí příkazu
      - **mono Mesta.exe**
- **Výsledky programu**
  - **Program dosahuje uspokojivých výsledků, kdy dosáhne nalezení nejkratší cesty pod 3000 km, kdy startuje i na hodnotách přes 7000. Kvalita závisí na hodnotě mutace zásadním způsobem, protože příliš vysoká hodnota degeneruje populaci a v momentě nalezení kvalitní Cesty, je zničena za pomoci mutace, jelikož je její dobrá souslednost zamění. Odzkoušená mutace je v rozsahu 1 – 0,5 %, kdy nedochází ke snižování kvality generace a současně program nekončí v lokálních maximech. Současně je zásadní velikost generace, jelikož při běhu programu bez mutace není možné nalézt dobré řešení bez dostatečného počtu členů v generaci. Při použití mutace se tento vliv snižuje, avšak je stále vhodné volit velikost populace min 100.**

Seznam částí programu

Třídy a jejich metody a atributy

1) **GenetickyAlgoritmus**
   a. **Metody**
      i. `GenetickyAlgoritmus()`
      ii. `void VytvorNovouGeneraci(int pocet)`
      iii. `void Mutace()`
      iv. `void VytvorPotomky(int velikostZacatku, int velikostKonce)`
      v. `void VypocitejKvalitu()`
      vi. `void VytvorZakladniGeneraci(int pocet)`
      vii. `void VykresliInfo(int generace,byte vstup)`
      viii. `string[] DefinujMapuVzdalenosti()`
      ix. `float[] SpocitejVzdalenosti(int id)`
   b. **Atributy**
      i. `float[,] mapaVzdalenosti`
      ii. `List<Mesto> města`
      iii. `List<Cesta> Generace`
      iv. `Cesta posledniCesta`
      v. `Random random`
      vi. `int IdCesta = 0`
      vii. `int IdGenerace = 0`
      viii. `Cesta nejlepsiCesta`
      ix. `const int PocetPotomku = 500`
2) **Cesta**
   a. **Metody**
      i. `Cesta(List<Mesto> list,int id)`
      ii. `override string ToString()`
   b. **Atributy**
      i. `int Id`
      ii. `List<Mesto> seznamMest`
      iii. `float Vzdalenost`
      iv. `int MinRozsah`
      v. `int MaxRozsah`
3) **Mesto**
   a. **Metody**
      i. `Mesto(int id, string nazev, float[] vzdalenosti)`
   b. **Atributy**
      i. `int Id`
      ii. `string Nazev`
      iii. `List<float> Vzdalenost`

Funkce programu

1) Deklarace **mapyVzdalenosti,** do které jsou uloženy vztažné vzdálenosti mezi všemi městy, tak aby bylo možné dále tyto informace využít. A následné zavolání metody **DefinujMapuVzdalenosti,** která do této matice 35 x 35 uloží všechny vztažné vzdálenosti. Současně tato metoda vrátí názvy všech měst ve formě pole stringů. Následně vytvořím základní skupinu 35 měst (**ukládám do kolekce Mesta**), kterým pomocí pole stringů minulé metody nastavím název a dále vzdálenosti do všech ostatních měst, pomocí metody **SpocitejVzdalenosti.** Tato metoda vybere ze základního pole **mapaVzdalenosti** vzdálenosti, které odpovídají danému městu. Veškeré návaznosti měst mezi sebou, jsou v programu řešeny pomocí **ID** u každého města. Tento jedinečný identifikátor, umožňuje snadnou práci s městy, aniž by nastala chyba například ve stanovení vzdáleností mezi dvěma městy.

```
453     public string[] DefinujMapuVzdalenosti()
```

```
496     private float[] SpocitejVzdalenosti(int id)
```

2) Vytvoří se základní generace potomků, pomocí metody **VytvorZakladniGeneraci**, která náhodně vygeneruje soustavu 35 měst, které se spojí do **CESTY** (jeden potomek Generace). Náhodné vytvoření je řešeno pomocí náhodného výběru ze seznamu čísel od 0 do 34, s tím, že každé tažené číslo se vymaže ze seznamu a tahá se další. Do té doby, dokud není nalezeno 35 měst, které se neopakují a tvoří jednu Cestu / potomka. Poté je tato **Cesta** přidána do generace, do té doby, dokud není vytvořena **Generace** o požadovaném počtu potomků. Město je vždy vytvořeno pomocí vzoru, který představuje kolekce **Mesta**, kde jsou v původním pořadí vytvořena města. Využije se tedy pouze informace z tohoto vzoru, která se předá konstruktoru nového města, aby vznikla nová instance. Výběr, jaké město je tvořeno je opět pomocí **ID**, které má každé z nich.

```
326     public void VytvorZakladniGeneraci(int pocet)
```

3) Zavolá se metoda **VytvorNovouGeneraci,** která sdružuje funkce genetického algoritmu do jediné metody. Uvnitř jsou poté volány jednotlivé části potřebné pro tento algoritmus. Nejprve se definuje pole hodnot od 5 do 25 se změnou o 5. Toto pole slouží pro změnu poměru informací braných od rodičů. Například se nastaví poměr 5 od prvního a 30 od druhého. Následuje cyklus, který začíná metodou **VypocitejKvalitu** (**viz. Ad 4**)**,** která představuje fitness funkci algoritmus. Následuje **VytvorPotomky** (**viz. Ad 5**), jež tvoří nové potomky při zadání poměru děděných informací od svých rodičů. A metodu uzavírá **Mutace (viz. Ad 6),** mutující jednotlivé potomky. Celý proces se poté opakuje dokud není dosaženo počtu generací zadaných na vstupu metody **VytvorNovouGeneraci.**

```
85      public void VytvorNovouGeneraci(int pocet)
```

4) **Vypocitej kvalitu –** nejprve je spočítán rozsah na vážené ruletě každého prvku **Cesta**. Tedy od, do zasashuje. Jakmile se takto projde celá generace a každý prvek **Cesta** má ve svých atributech uloženou informaci o minimálním a maximálním rozsahu. Poté se ,,roztočí,, vážená ruleta a náhodně se vybírá číslo od 0 do 5000, což je stanovený rozsah pro nastavenou generaci. Toto náhodné číslo se poté kontroluje s rozsahem každého prvku v generaci. Jakmile je zjištěno, že prvek má rozsah, který obsahuje náhodné číslo, tak se tento prvek **Cesta** vybere a uloží do pomocné kolekce. Tento výběr se provádí do té doby, dokud nemá generace požadovaný počet potomků. Poté se předá reference na tuto kolekci původní

```
277        public void VypocitejKvalitu()
```

5) **VytvorPotomky –** provádí křížení dvou prvků Generace (**Cesta1 a Cesta2**) v zadaném poměru od 5 ku 30 až do 30 ku 5 pro prvního potomka. Vytvoření nového potomka probíhá tak, že se do nového uloží všech 35 měst prvního rodiče, poté se smaže část, která má patřit druhému rodiči. Následně se přiloží 35 měst z druhého rodiče a smaže se první část z těchto nových 35, tak že výsledný potomek obsahuje 35 měst v kombinaci od obou rodičů. Samozřejmě každý z potomků obsahuje některá města dvakrát, a proto je nutné tyto duplicity odstranit. Toto odstranění je provedeno tak, že první potomek obsahuje prvky, které jsou u něj duplicitní, ale scházejí druhému potomku. Proto se pouze duplicity prohodí mezi potomky, a vzniknou dva potomci bez duplicit. Tyto potomky poté uložím do kolekce **Generace**.

```
154        public void VytvorPotomky(int velikostZacatku, int velikostKonce
```

6) **Mutace –** prochází se všechna **Města v Cesta,** dokud se nenalezne 400. prvek této řady. Jakmile se tento prvek (**Mesto**) nalezne, prohodí se s některým v dané cestě. Tím vznikne náhodná mutace, protože se náhodně změní prvek, aniž by k tomu byl jiný důvod. Tento krok zamezuje, aby program neskončil na lokálním maximu a přiblížil se skutečnému maximu. Mutace je nastavena na 0,5 %, protože při vyšších hodnotách měl program problém se dostat ke kratším cestám. Nalezl je, avšak trvalo to o dost déle. Takto nízká mutace probouzí generaci, avšak neníčí ji přiliš častými změnami.

```
109    public void Mutace()
```

Pro vyzkoušení je možné tuto mutaci změnit a nastavit jinou hodnot než výše zmíněné půl procento. Stačí zaměnit číslo po celočíselném dělení viz. Obrázek.

```
120            if (cisloPrvku%400 == 0)
```