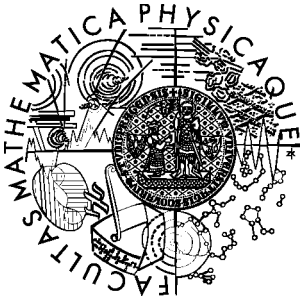
Univerzita Karlova v Prahe
Matematicko-fyzikálna fakulta

BAKALÁRSKA PRÁCA




Martin Straka

# Šírenie meškania na železničnej sieti.

Katedra aplikovanej matematiky

Vedúci bakalárskej práce: Mgr. Robert Babilon

Študijný program: Informatika, Programovanie

2009

Ďakujem Mgr. Robertovi Babilonovi, vedúcemu mojej bakalárskej práce, za jeho pomoc, cenné rady, odporučenia a zhovievavosť pri tvorbe práce,
Českým dráham za poskytnutie dát, výpravcom Pavlovi Jakubíkovi a Jaroslavovi Tasarymu a všetkým ľudom z oblastného riaditeľstva ŽSR v Trnave, že si našli na mňa čas a poskytli mi svoje dlhoročné praktické skúsenosti.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

V Prahe dňa Martin Straka

# Obsah

Názov práce: Šírenie meškania na železničnej sieti.
Autor: Martin Straka
Katedra (ústav): Katedra aplikovanej matematiky
Vedúci bakalárskej práce: Mgr. Robert Babilon
e-mail vedúceho: babilon@kam.ms.mff.cuni.cz

Abstrakt: Predmetom tejto práce je simulácia šírenia meškania vlakov na železničnej sieti. Program graficky ukazuje jednotlivé simulácie a tiež vytvára ich štatistiky pri rôznych kritériách (napr. meniace sa čakacie časy alebo zmeny križovaní jednotlivých vlakov). Práca tieto výsledky následne vyhodnocuje a porovnáva. Je tu tiež snaha navrhnúť vhodnú voľbu takých kritérií, aby vlaky meškali čo najmenej. Súčasťou práce je tiež vyhľadávanie slabých miest v grafikone. Teda vlakov, na ktoré sa šíri meškanie najčastejšie, alebo ktoré ako prípoje nečakajú na ich predchádzajúce vlaky z dôvodu veľkého meškania.

Kľúčová slová: vlaky, meškanie, simulácia, štatistiky

Title: Spread of delay on railway network
Author: Martin Straka
Department: Department of Applied Mathematics
Supervisor: Mgr. Robert Babilon
Supervisor's e-mail address: babilon@kam.ms.mff.cuni.cz

Abstract: The subject of this work is a simulation of late trains spreading in railway net. This program shows the single simulations and creates their statistics according to different criteria (for example changing waiting periods or changes in crossing of the trains). This work evaluates and compares the results. Next point of this work is an effort to suggest the most appropriate criteria in order to prevent the train lag. Another element of this work includes searching for weak points in grafikon, which includes the trains that are late more frequently than other or do not wait for an arrival of the previous ones on their way because of the great lag. Keywords: trains, late, simulation, statistic

Keywords: trains, late, simulation, statistic

# 1 Úvod

Každý z nás asi pozná nepríjemné situácie na železnici, keď vlaky meškajú alebo nám ujde prípoj, na ktorý sme chceli ísť. Z tohto vzniká otázka, či vlaky musia skutočne meškať tak veľa alebo tak často. Nedajú sa vymyslieť efektívnejšie pravidlá pre odstraňovanie meškania, ako sú tie súčasné?

Podľa predpisov ŽSR ([3]) sa meškanie môže znížiť krátením pobytov v stanici a využitím technických možností vlaku, nesmie sa prenášať na iné vlaky a musí sa upraviť križovanie tak, aby dôležitejší vlak nebol zmeškaný menej dôležitým.

V praxi však o všetkom rozhodujú výpravcovia a vlakoví dispečeri. Pri križovaní vlakov s rovnakou prioritou sa obvykle volí taká varianta, aby súčet meškania oboch vlakov bol čo najmenší. Ak by sa malo meškanie preniesť na vlak s väčšou prioritou, tak tu si už musí výpravca pýtať povolenie od vlakového dispečera. Ďalej sa pri rozhodovaní zohľadňuje aj veľa iných faktorov. Napr. : počet prestupujúcich cestujúcich, dôležitosť ďalších vlakov na ceste, ...

U ČD nejaké presné pravidla nie sú určené, len sa hovorí, že by sa mali minimalizovať dopady meškania.

V práci je implementovaný program pre grafické znázornenie šírenia meškania vlakov v železničnej sieti, vyhodnocovanie štatistík z výsledkov týchto simulácií a vyhľadávanie vlakov, ktoré majú tendenciu stať sa najčastejšie zmeškanými alebo ujdenými. Práca potom porovnáva výsledky pri rôznych počiatočných podmienkach a pravidlách meškania.

## 1.1 Rozdelenie práce

Celá práca sa skladá zo 5 hlavných častí:

- Vyber vhodných prostriedkov pre aplikáciu
- Krátky popis aplikácie
- Vytvorenie vstupných dát pre hlavný program simulácie
- Simulácia meškania
- Vyhodnotenie výsledkov

Prvá časť sa zaoberá voľbou platformy, programovacieho jazyka a grafického prostredia.

Druhá časť stručne popisuje získané vstupné dáta, ich formát pre hlavný program simulácie a algoritmus riešenia.

Tretia časť pozostáva z návrhu rýchlej reprezentácie výsledného vstupného súboru pre simuláciu a implementáciu jeho vytvorenia zo vstupných dát od ČD a Mgr. Roberta Babilona. Jej veľká náročnosť spočívala vo veľkosti a nekorektnosti vstupných dát od Českých dráh. I keď sa jedná len o jedno spustenie pred hlavným programom, tak pre rozumnú výslednú rýchlosť tu museli byť zahrnuté efektívne algoritmy výpočtu.

Štvrtá časť pozostáva z opisu inštalácie programu, vstupných dát a dátových štruktúr, užívateľskej a programátorskej dokumentácie aplikácie.

Piata časť vyhodnocuje a porovnáva výsledky zo štatistík.

V nasledujúcich kapitolách rozoberiem jednotlivé časti podrobnejšie.

# 2 Výber prostriedkov pre aplikáciu

## 2.1 Výber programovacieho jazyka

Vzhľadom k pomerne veľkej výpočetnej náročnosti programu bolo potrebné zvoliť rýchly programovací jazyk. Je to úloha z praxe, ktorá je veľmi dobre implementovateľná pomocou objektov. Po zvážení týchto faktorov som sa rozhodol pre jazyk C++.

## 2.2 Grafika

Z výberu som vylúčil knižnice, ktorých licencia by bránila voľnému použitiu a tie, ktoré sú zamerané len na jednu platformu. Po splnení týchto požiadaviek ostali 4 vhodné a známe knižnice pre ktoré existuje aplikačné rozhranie v C++: SDL, Opengl, wxWidgets a GTK+. Z dôvodu výbornej prenositeľnosti som sa nakoniec rozhodol pre SDL s použitím knižníc SDL, SDLmain, SDL_image a SDL_ttf, ktoré sú voľne stiahnuteľné z http://www.libsdl.org/.

## 2.3 Výber vývojového prostredia

Pôvodne som projekt robil vo Microsoft Visual studio 2008, ale pre veľké problémy s prenositeľnosťou na iné počítače som nakoniec prešiel na MinGW Developer studio, ktoré využívajú prekladač gcc a sú multiplatformové. Ich ďalšou výhodou je menšia náročnosť (pamäťová i výpočtová) na počítač. Zaujímavosťou je, že počítanie štatistík môjho programu beží znateľne rýchlejšie pod MingGW ako pod Visual Studiom.

# 3 Krátky opis fungovania programu

Pre to, aby sme mohli správne pochopiť výsledky zo simulácií, potrebujeme vedieť ako program funguje vzhľadom na svoje vstupné dáta.

## 3.1 Vstupné dáta

Niektoré dôležité informácie o vlakoch a ich cestách sa nepodarilo získať od ČD, tak sú v programe nastavené defaultne ako konštanty alebo vypočítané podľa niektorých kritérií. Podľa priority stanice (1,2,3) sú nastavené nasledujúce konštanty :

počet koľají stanice1 (3,10,15), čas potrebný na vystúpenie a nastúpenie cestujúcich (1,2,4) a čas potrebný na prestup v stanici (0,2,4).

Čas na postavenie vlakovej cesty je nastavený na hodnotu 1. O koľko je vlak schopný ísť rýchlejšie medzi susednými stanicami je určené na hodnotu 7% z celkovej doby jazdy vlaku na tomto úseku.

Najdôležitejším vstupným údajom programu, ktorý sa nepodarilo získať je zoznam prípojov pre jednotlivé vlaky. Tento zoznam je určený tak, že do neho patria všetky vlaky, ktoré odchádzajú do pol hodiny (konštanta) zo stanice príchodu daného vlaku.

Ostatné dáta, ktorými sú stanice, ich súradnice, vlaky, ich príchody a odchody, dni, kedy chodia a počet koľají medzi stanicami sa podarilo získať z bežnej praxe z grafikonu z roku 2007 a ďalších pomocných zdrojov od Mgr. R. Babilona.

### 3.1.1 Formát vstupných dát pre simuláciu

Najdôležitejším požiadavkom na formát vstupných dát je čo najrýchlejšie nájdenie ľubovoľného vlaku v nich. Preto bol zvolený index-sekvenčný súbor v adresári „Vlaky“. Jeden vlak v týchto vstupných dátach vyzerá napr. takto:

```
Os|05172/20054|
Os|50|1|3|12|1|1|0|5453840|577|133|Náchod|1|07|58|
Os|50|1|3|8|1|1|0|5453860|580|130|Náchod-Běloves z|1|08|01|1|08|01|05163/30054|
Os|50|1|3|8|1|1|0|5453850|580|127|Náchod-Malé Poříčí z|1|08|03|1|08|03|05163/30054|
Os|50|1|3|12|1|1|0|5453851|579|124|Velké Poříčí z|1|08|05|1|08|05|05163/30054|
Os|50|1|3|1|1|1|0|5453820|578|120|Hronov|1|08|08|05163/30054|
```

Prvý riadok tvorí typ vlaku a jeho poradové číslo.
Druhý riadok tvorí štartovná stanica vlaku. Sú tam po rade tieto položky: typ vlaku v tejto stanici (môže sa meniť počas jazdy vlaku), označenie v ktorý deň vlak chodí, počet koľají medzi touto stanicou a nasledujúcou, počet koľají v stanici, údaj o koľko je vlak schopný ísť rýchlejšie medzi touto stanicou a ďalšou (defaultne nastavené na 7% doby jazdy vlaku v sekundách), čas na postavenie vlakovej cesty, čas potrebný na vystúpenie a nastúpenie cestujúcich, čas potrebný na prestup v stanici, identifikátor stanice, jej x-ová súradnica, y-ová súradnica, názov stanice, čas odchodu (tri položky: deň, hodina, minúta)
Tretí riadok je to isté ako druhý s tým rozdielom, že posledné tri položky z druhého riadku označujú príchod do stanice, ďalšie tri odchod a potom už nasleduje zoznam možných prípojov. V druhom riadku tento zoznam nikdy nie je, pretože vlak nemôže prísť do svojej prvej stanice s meškaním. Už tam je a určite tam na neho nečakajú žiadne vlaky.
Posledná stanica je taká istá ako tretí riadok akurát s tým rozdielom, že tam chýbajú údaje o odchode vlaku a niektoré údaje sú tam už navyše ako nepotrebné konštanty. Je to praktické z toho dôvodu, aby sa ušetrilo dosť podmienok v hlavnom programe (väčšina položiek je stále na tých istých pozíciách).

## 3.2 Beh programu

Užívateľ najprv určí, či chce generovať štatistiky alebo si chce pozrieť simuláciu.

### 3.2.1 Simulácia

Pri tejto možnosti následne zadá dátum, časový krok a jednu z variant pravidiel, podľa ktorých budú vlaky na seba čakať . Potom môže pridať do simulácie ľubovoľne veľa vlakov. Každý z nich bude určený číslom vlaku, stanicou do ktorej príde s oneskorením (napr. dôsledkom havárie, technickej chyby a pod.).) a veľkosťou meškania. Čas v simulácii je určený prvým vlakom (dátumom, stanicou a meškaním). Všetky meškajúce vlaky sú pridané do fronty. Po zadaní každého z nich sa hľadajú ich prípoje a tie sú tiež vložené do fronty. Pri pridávaní vlakov do fronty môžu nastať štyri varianty (ozn.: 'vlak 1' je vlak, ktorý sa pridáva do fronty) :

1. Pridávaný vlak ešte vo fronte nie je. *Pridá sa na koniec*
2. Pridávaný vlak už vo fronte je (rovnaké číslo vlaku ako nejaký vlak z fronty), má rovnakú pozíciu (aktuálna stanica jazdy vlaku) ako ten vlak vo fronte a väčšie meškanie. Jedná sa o ten istý vlak, ktorý vo fronte už je (platí to aj v ďalších variantách), len pridávaný prípoj má väčšie meškanie.
Riešenie: *Vlaku vo fronte sa priradí meškanie pridávaného vlaku do fronty.*
Je to z toho dôvodu, že tento vlak tu musí čakať ešte aj na nejaký iný prípoj (viac omeškaný) ako pôvodne.
3. Pridávaný vlak tam už je a má menšiu pozíciu ako vlak vo fronte.
Riešenie: *Vlaku vo fronte sa priradí pozícia pridávaného vlaku a meškanie na tejto pozícii. Meškanie na pôvodnej pozícii ostane zachované. Opäť sa prepočíta jazda vlaku až do času v simulácii. Počas tohto výpočtu bude uprednostňované vyššie*

*meškanie vo vektore meškaní vlaku.*
4. Pridávaný vlak tam už je a má väčšiu pozíciu ako vlak vo fronte.
Riešenie: *Vlaku vo fronte sa priradí meškanie pridávaného vlaku na príslušnú budúcu pozíciu. A potom pri jazde vlaku sa porovná, či tam už nie je väčšie meškanie. Ak áno, tak sa ponechá a neznižuje sa.*

V každom časovom kroku simulácie sa spustí cyklus cez všetky vlaky vo fronte až dovtedy, kým všetky vlaky vo fronte (aj novo pridané počas tohto kroku) nie sú prepočítané. Počas výpočtu jedného vlaku program posunie vlak dopredu o maximum staníc tak, aby sa neprekročil časový krok simulácie. Zároveň sa počíta jeho nové meškanie v každej stanici. Znižuje sa o zrýchlenie medzi stanicami vďaka technickým možnostiam vlaku a ak je to možné, tak aj krátením pobytu v stanici. Ak sa meškanie stratilo, alebo vlak prišiel do svojej konečnej stanice, tak sa vlak vyhodí z fronty. Každý posun a jednu stanicu sa vykresľuje na mapu a zároveň sa prerátavajú možné prípoje daného vlaku. Buď sa pridávajú do fronty meškajúcich vlakov alebo program vyhodnotí, že ich nechá odísť a nebudú čakať na vlak 1. Toto rozhodnutie prebieha na základe mnohých podmienok: dátum, čas, varianta pravidiel meškania, priorita vlaku, spôsob križovania, počet koľají medzi stanicami, čakacie doby na prestup, nástup a výstup cestujúcich v stanici, zrýchlenie vlaku vďaka jeho technickým možnostiam vlaku a krátením pobytu v stanici.

Po každom časovom kroku užívateľ môže zmeniť krokovanie simulácie alebo pridať ďalší vlak. Ovládanie je priamo v programe.

Program končí, keď je fronta prázdna.

### 3.2.2. Štatistiky

Najprv sa vygeneruje vektor 350 náhodných čísiel z intervalu 1-364 bez opakovania. To budú dni. Následne užívateľ zadá počet vlakov, pre ktoré chce, aby sa počítali štatistiky pre všetky varianty. Potom sa zo zoznamu všetkých vlakov postupne vyberá každý dvadsiaty štvrtý. Je to preto, aby sa približne pokrylo celé rozpätie čísiel vlakov. Tých je 8624 (350*24=8400). Pre každý vybratý vlak sa počíta simulácia popísaná v predchádzajúcej kapitole. Stanica do ktorej príde vlak s meškaním bude defaultne nastavená na druhú s smere jazdy vlaku

Výsledky z týchto simulácii pre jednotlivé varianty sú ukladané do súboroch vlaky_var[1-9]_[uj|zm].txt a suhrn_vysledkov.txt.

# 4 Vytvorenie vstupných dát pre hlavný program simulácie

## 4.1 Vstupy a výstupy

Dodané vstupné súbory od ČD sú „VLAK_CEV.txt“ a „KVL_BIN.txt“. Od Mgr. Roberta Babilona sú „dvojkolej.txt“ a „suradnice.txt“.
Ďalšie vytvorené pomocné vstupné súbory: stanice.txt, dvoj_uprav.txt, subor1.txt, subor2.txt, subor3.txt, subor3_upraveny.txt, subor4.txt, vlaky.txt.
Výstupným súborom je “vlaky.txt” v jeho index - sekvenčnej podobe v adresári “Vlaky”, ktorý slúži ako základná dátová štruktúra pre hlavný program simulácie.
Všetky súbory vrátane prevodného programu prikladám v CD v adresári „Vstupne data“.

### 4.1.1 Popis súboru „VLAK_CEV.txt“

Nasledujúci popis je od ČD:
*VLAK_CEV.TXT je textový soubor v kódování ANSI 1250, který obsahuje informace o druhu vlaku a trase vlaku. Každý jednotlivý záznam se skládá z hlavičky, popisu trasy vlaku v chronologickém sledu, fakultativně je dále uváděna informace o službách na vlaku a též název vlaku. Soubor obsahuje i některé z údajů, které nejspíš nelze jednoduše strojově zpracovávat. Rád bych doplnil, že ty slouží výlučně ke zběžné kontrole takto připravovaných dat a nepředpokládám, že byste se jimi zabýval.*
*V hlavičce vlaku je na pozici 1 - 4 kategorie vlaku, dále na pozici 5 – 11 výchozí stanice, 14 - 20 cílová stanice, počínaje pozicí 22 je uvedena trasa vlaku textově.*

*V hlavičce vlaku může být několik záznamů o kategorii.Dále je v hlavičce uvedeno ve větě se řetězcem "Aktualizace" (pozice 1 - 11) datum poslední úpravy dat o vlaku v CEV, a sice 13 - 14 den, 16 – 17 měsíc, 19 - 22 rok, na pozicích 29 - 36 a 38 - 45 je ve tvaru DDMMRRRR uvedena platnost dat konkrétního vlaku. Na pozici 67 - 77 je ID vlaku.*

*Záznam o trase vlaku obsahuje tyto položky:*

*Na pozici 1 - 7 číslo vlaku včetně lomení (pouze u výchozí stanice), 9 - 15 číslo stanice, 17 - 26 název stanice, 40 den příjezdu vlaku do stanice, 42 - 43 hodina příjezdu vlaku do stanice, 45 - 46 minuta příjezdu vlaku do stanice, 50 den odjezdu vlaku ze stanice, 52 - 53 hodina odjezdu vlaku ze stanice, 55 - 56 minuta odjezdu vlaku ze stanice, 58 - 62 aktuální číslo vlaku, 74 - 77 číslo kalendáře vlaku, který platí z dané stanice (viz KVL.TXT nebo KVL_BIN.TXT), 79 - 82 číslo kalendáře vlaku, který platí pro omezené zastavení vlaku v dané stanici (viz KVL.TXT nebo KVL_BIN.TXT).*

*Položky na pozicích 64, 66, 68, 70, 72, 84 a 86:*

*Je-li na pozici 64 "1", vlak v dané stanici/zastávce zastavuje na znamení.*

*Je-li na pozici 66 "1", vlak v dané stanici/zastávce zastavuje pouze ve dnech podle kalendáře s číslem na pozicích 79 - 82.*

*Je-li na pozici 68 "1", vlak v dané stanici/zastávce zastavuje výlučně pro nástup cestujících.*

*Je-li na pozici 70 "1", vlak v dané stanici/zastávce zastavuje výlučně pro výstup cestujících.*

*Je-li na pozici 72 "1", vlak v dané stanici/zastávce zastavuje jen z dopravních důvodů; Informace o zastavování tudíž není určena pro cestující.*

*Je-li na pozici 84 "1", vlak může odjet ihned po ukončení výstupu cestujících (viz symbol obráceného písmene T v KJŘ).*

*Je-li na pozici 86 "1", odjezd vlaku do následující stanice je možný již v čase příjezdu.*

*Pod záznamy o trase vlaku je pro kontrolu text kalendáře (více kalendářů).*

Ukážka jedného vlaku v popisovanom súbore (font zmenšený kôli zachovaniu riadkov ) :

---

```
Os  5434144->5434804 (Nový Jičín město->Suchdol nad Odrou)
Aktualizace:16.04.2007 pro: 10062007-08122007                    13353/30054
13353/3 5434144 Nový Jičín město      | |  |  | |1|04|41|13353| | | | | |  85|    | | |
        5434154 Nový Jičín zast. nz   |1|04|44| |1|04|44|13353|1| | | | |    |    | | |
        5434804 Suchdol nad Odrou     |1|04|55| | |  |  |     | | | | | |    |    | | |
KVL:0085 jede v x,nejede 27.-29.XII.|
OBP:10160 5434144->5434804 10043
samoobslužný (specifický) způsob odbavení cestujících (Nový Jičín město->Suchdol nad
Odrou); kalend.:10043
```

---

## 4.1.2 Popis súboru „KVL_BIN.txt“

Každý riadok súboru „KVL_BIN.txt“ pozostáva z dvoch častí. Prvá je identifikátor a druhá je zoznam jednotiek (vlak v daný deň chodí ) a núl (nechodí) . Dni začínajú od 10.12.2006.

Napr. druhá jednotka alebo nula označuje, že daný vlak 11.12.2006 chodí.

V programe simulácie je tento súbor premenovaný na „dni.txt“.

## 4.1.3 Popis súboru „dvojkolej.txt“

V súbore „dvojkolej.txt“ sú dvojkoľajové trate v ČR. Každá je oddelená dvoma riadkami, ktoré začínajú „*“. Medzi nimi je zoznam po sebe idúcich staníc (tri položky: prvá je nepotrebná, identifikátor stanice, meno stanice) .

Ukážka:

```
********** 091a
  0 5457076 Praha hl.n.
  3 5457256 Praha-Holešovice
********** 130, časť 090
```

### 4.1.4 Popis súboru „suradnice.txt“

V každom riadku súboru „suradnice.txt“ sú 4 položky oddelené medzerami: identifikátor stanice, jej x - ová súradnica, y - ová súradnica a jej názov. Prevod na pixely sa vykonáva pomocou nasledujúcich funkcií:

```
int Xm (int x) {return (int)(0.1438* (x-12156));}
int Ym (int y) {return (int)(0.2236* (51012-y));}
```

### 4.1.5 Popis súboru „stanice.txt“

V súbore „stanice.txt“ sú všetky stanice zo súboru „VLAK_CEV.txt“. V každom riadku sú tri položky:
Identifikátor stanice, názov stanice, jej typ (1,2 alebo 3) .

Typ 1: Cez stanicu prechádzajú iba Os a Sp vlaky
Typ 2: Cez stanicu prechádza aspoň jeden R alebo Ex.
Typ 3: Cez stanicu prechádza aspoň jeden vlak s vyššou prioritou ako 2 (IC,EC,EN,SC) .

## 4.2 Reprezentácia dát

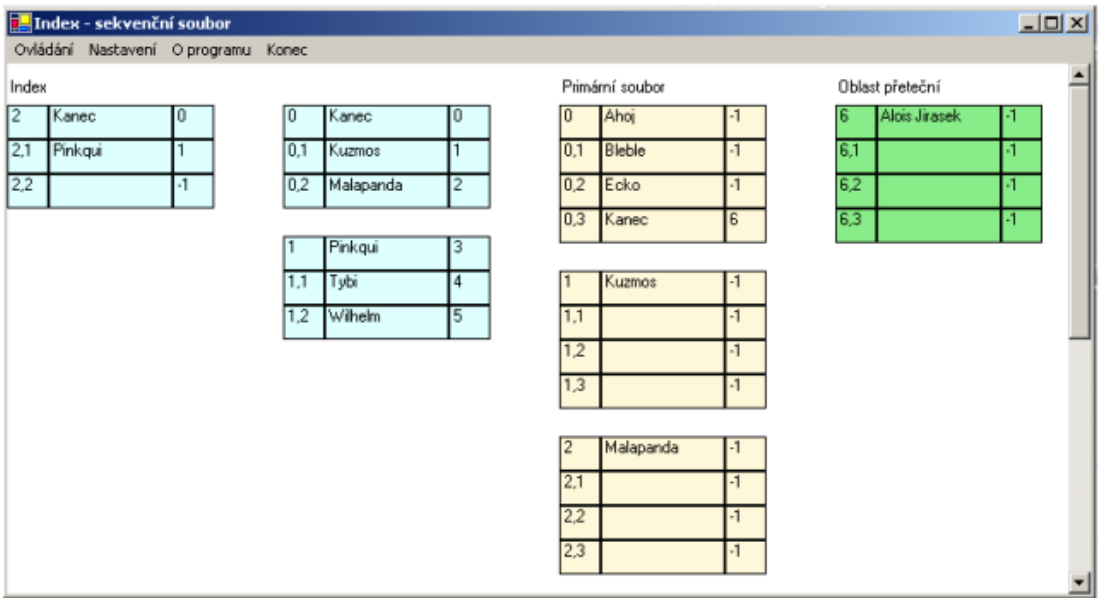
Z dôvodu, že táto časť programu bude spustená len jeden krát, nebolo potrebné, aby bežala úplne najrýchlejšie. Avšak vzhľadom k veľkosti vstupných (VLAK_CEV.txt má cca 200 000 riadkov), kde bude potrebné často vyhľadávať, bolo potrebné aspoň u niektorých súborov zvoliť vhodnejšiu reprezentáciu ako len prístup po jednotlivých riadkoch. Vybral som prispôsobený index - sekvenčný súbor s úrovňou indexu 1, čím sa program výrazne urýchlil. Jednotlivé podrobnosti implementácie sú v kapitolách 4.3.5 a 4.3.9.

### 4.2.1 Index sekvenčný súbor

Táto štruktúra je navrhnutá pre efektívny náhodný prístup k jej jednotlivým položkám. Skladá sa z primárneho súboru zotriedeného podľa kľúča a obecne s viacúrovňovou štruktúrou indexu. Index je B-strom odkazujúci na jednotlivé časti primárneho súboru.

Pomocou indexu sa do súboru môžu pridávať alebo odoberať prvky. Ak sa prvok nezmestí do stránky v primárneho súbore, dá sa do oblasti pretečenia. Ak je príliš veľa prvkov v oblasti pretečenia, alebo sú jednotlivé stránky primárneho súboru málo zaplnené, je možné previesť reorganizáciu- Presypanie oblasti pretečenia do primárneho súboru a vytvorenie nového indexu. Tým sa zefektívni vyhľadávanie.

Príklad:




Obr. 1 Index-sekvenčný súbor

Plocha okna obsahuje farebné obdĺžniky, reprezentujúce jednotlivé stránky. V každej z nich je niekoľko záznamov. Ich počet je určený **blokovacím faktorom b**.
Záznam je rozdelený do troch častí - číslo v ľavej časti je **adresa záznamu**. Napr. 1,3 znamená, že záznam sa nachádza na prvej stránke s offsetem 3 (je teda štvrtý od začiatku).
Stredná časť obsahuje jeho **hodnotu**- textový reťazec. Podľa nej sa vyhľadáva v indexe a zároveň je to triediaci kľúč v primárnom súbore.
Pravá časť obsahuje **odkaz** na ďalšiu stránku nižšej úrovne. U primárneho súboru má buď ukončovaciu hodnotu (na obr. -1) alebo ukazuje na stránku pretečenia, pretože daný záznam sa nezmestil už do jemu prislúchajúcej stránky.

## 4.3 Popis implementácie

### 4.3.1 Funkcia „subor_dvojkolaj()“

Funkcia „subor_dvojkolaj()“ upraví súbor „dvojkolaj.txt“ tak, že na výstupe do súboru „dvoj_uprav.txt“ budú v jednom riadku mestá, ktoré sú spojené dvojkoľajovou traťou.

### 4.3.2 Funkcia „subor1()“

Funkcia „subor1()“ vytiahne zo súboru „VLAK_CEV.txt“ potrebné údaje a pridá ďalšie podľa priority stanice, kedy vlak chodí a o akú trať sa jedná (dvoj alebo jednokoľajová). Následne zoradí výsledok podľa čísla vlaku do „subor1.txt“ .

**Popis operácii funkcie:**
- vyhodí z „VLAK_CEV.txt“ všetky vlaky, ktoré nie sú osobné. Taktiež všetky, ktoré neprechádzajú ani jednou stanicou v ČR.

-zo súboru „KVL_BIN.txt“ pridá číslo, kedy vlak chodí. (Možné urýchlenie zmenou reprezentácie „KVL_BIN.txt“ ) .
-zo súboru „dvoj_uprav.txt“ určí, či sa jedná o dvojkoľajovú alebo len jednokoľajovú trať pomocou funkcie “je_dvojkolaj()” .
-zo „stanice.txt“ určí aký je to typ vlaku (funkcia “aka_stanica()” ) a pridá podľa toho konštanty pomocou funkcie “cakacie_doby()”. (Možné urýchlenie zmenou reprezentácie stanice.txt)
Keď je v jednom vlaku (jeho čísle) aj nákladný aj osobný vlak, tak vytiahne z neho len ten osobný.

**Popis jednotlivých riadkov a ich položiek v „subor1.txt“ oddelené „|“**

1. riadok: Typ vlaku, jeho identifikátor
2. riadok:
1. položka: Typ vlaku odchádzajúceho zo stanice na tomto riadku.
2. položka: Číslo kedy vlak chodí z KVL_BIN.
3. položka: Počet koľají medzi stanicou na tomto riadku a ďalšou.
   V poslednom riadku je tento údaj navyše z dôvodu zjednodušenia programu.
4. položka: Počet koľají stanice1 (3,10,15). Podľa priority.
5. položka: O koľko je vlak schopný ísť rýchlejšie medzi stanicou na tomto riadku a ďalšou (defaultne 7% doby jazdy vlaku).
6. položka: Čas na postavenie vlakovej cesty (1).
7. položka: Čas potrebný na vystúpenie a nastúpenie cestujúcich (1,2,4).
8. položka: Čas potrebný na prestup v stanici (0,2,4).
9. položka: Identifikátor (číslo) stanice.
10. položka: Súradnica x.
11. položka: Súradnica y.
12. položka: Názov stanice (string).

13. položka: Príchod (deň).

14. položka: Príchod (hodina).

15. položka: Príchod (minúta).

16. položka: Odchod (deň).

17. položka: Odchod (hodina).

18. položka: Odchod (minúta)

**Ukážka jedného vlaku:**

```
EC|00070/00054|
EC|1|1|15|1|1|4|4|8102852|Wien Südbahnhof|1|18|33|
EC|1|1|15|1|1|4|4|8102825|Hohenau|1|19|19|1|19|20|
EC|1|2|15|1|1|4|4|5433425|Břeclav os.n.|1|19|33|1|19|47|
EC|1|2|15|1|1|4|4|5433295|Brno hl.n.|1|20|18|1|20|20|
EC|1|2|15|1|1|4|4|5453913|Česká Třebová|1|21|21|1|21|22|
EC|1|2|15|1|1|4|4|5453613|Pardubice hl.n.|1|21|55|1|21|56|
EC|1|1|15|1|1|4|4|5453414|Kolín|1|22|16|1|22|17|
EC|1|1|15|1|1|4|4|5457076|Praha hl.n.|1|22|58|
```

Úmyselne je navrhnuté také rozloženie, aby čo najviac položiek malo nemenný index. Ušetria sa tým možné zbytočné podmienky v ďalšom behu programu.

### 4.3.3 Funkcia subor3()

Funkcia subor3() zotriedi „suradnice miest.txt“ podľa názvu stanice pre ľakšie vyhodenie problémových staníc s nekompletnými datami.
Vytvori subor3.txt vo formate:
Nazov stanice|cislo_stanice x_suradnica y_suradnica
Ukážka: Beroun ser.n.|78000 14059 49950

### 4.3.4. Funkcia stanice2()

Funkcia stanice2() postupne berie každú stanicu zo „stanice.txt“, pripája k nej jej súradnice zo „subor3_upraveny.txt“ a zároveň hľadá v „subor1.txt“ riadky, kde sa nachádza daná stanica.
Vyhodí riadky, kde táto stanica je konečná, zotriedi ich podľa času odchodu (u poslednej žiaden nie je) a pošle na výstup do „subor2.txt“.

**Popis jednotlivých riadkov a ich položiek v „subor2.txt“**

Jenodlivé položky súboru „subor2.txt“ sú oddelené znakom „|“ .

1. riadok: Identifikátor stanice, x-ová súradnica, y-ová súradnica, názov stanice, jej typ
2. riadok: Identifikátor vlaku, typ vlaku, hodina odchodu, minúta odchodu, kedy chodí (V „KVL_BIN.txt“)

**Ukážka jednej stanice:**

```
5454056|14217|50109|Dobrovíz/Dobrovíz z|1|
19700/00054|Os|09|55|Středokluky|298|
19701/10054|Os|12|18|Hostouň u Prahy z|298|
19702/20054|Os|14|17|Středokluky|298|
19703/30054|Os|17|49|Hostouň u Prahy z|298|
```

### 4.3.5 Funkcia stanica2_rozdelenie()

Vo funkcii „subor4()“ bude potrebné veľa krát vyhladávať stanice zo súboru „subor2.txt“. Bolo potrebné zvoliť vhodnejší spôsob, ako len sekvenčné prechádzanie. Kôli optimalizácii rýchlosti nájdenia konkrétnej stanice mení funkcia

„stanica2_rozdelenie()“ súbor „subor2.txt“ na index- sekvenčný a ukladá ho do adresáru „Subor2“ .
Realizuje to tak, že prepisuje „subor2.txt“ do menších súborov „subor2_[n]” , kde „n” je poradové číslo. Keď počet riadkov v „subor2_[n]” je konštatna „riadky=300“, dokončí zápis aktuálnej stanice, uzavrie súbor a pokračuje v súbore „subor2_[n+1]” až do konca „subor2.txt” .
Súčastne vytvára súbor „index.txt”, kde v každom riadku je identifikátor počiatočnej stanice zo súborov „subor2_[n]” a druhou položkou je jeho poradové čislo.
Pri tejto reprezentácii dokáže program nájsť hľadanú stanicu prejdením dvoch krátkych súborov, čo je výrazne rýchlejšie, ako len sekvenčné prechádzanie obrovského „subor2.txt“.

### 4.3.6 Funkcia subor4()

Funkcia „subor4()“ zoberie každý riadok zo „subor1.txt“ a ak je to podtrebné, tak k danej stanici v ceste vlaku pridá zoznam prípojov, ktoré možu čakať (const c_cakanie=30) na príchod vlaku v stanici z adresáru „Subor2“. Je tu ošetrený aj prechod cez dni. Napr. keď vlak príde 23:50 a čakanie je 30 minút, tak do subor4.txt sú pridané aj vlaky od polnoci do 0:20.
Výstup je súbor „subor4.txt“.

**Popis jednotlivých riadkov a ich položiek v „subor4.txt“**

1. riadok: Identifikátor vlaku, jeho typ, znak „@“ (len kôli rozpoznaniu riadku)
2. riadok: začiatočná stanica, súradnica x, súradnica y, nemá žiadne prípoje
3. riadok: začiatočná stanica, súradnica x, súradnica y, môže mať lubovolný počet prípojov

**Ukážka jedného vlaku:**

```
00070/00054|EC|@|
8102852|x|x|
8102825|x|x|
5433425|16895|48755|02331/10081|02029/90054|10126/60054
5433295|16622|49184|04119/90054|04619/90054|04089/90054|
04854/40054|04416/60054
```

### 4.3.7 Fukcia spojenie1a4()

Funkcia „spojenie1a4“ spojí „subor1.txt“ a „subor2.txt“ pričom prepočíta súradnice miest na pixely a výsledok vráti ako súbor „vlaky.txt“ . Možné prípoje dá na koniec jednotlivých riadkov.

**Ukážka jedného vlaku:**

```
EC|00070/00054|
EC|1|1|15|1|1|4|4|8102852|x|x|Wien Südbahnhof|1|18|33|
EC|1|1|15|1|1|4|4|8102825|x|x|Hohenau|1|19|19|1|19|20|
EC|1|2|15|1|1|4|4|5433425|681|504|Břeclav os.n.|1|19|33|1|19|
47|02331/10081|02029/90054|10126/60054|10149/90054|
EC|1|2|15|1|1|4|4|5433295|642|408|Brno hl.n.|1|20|18|1|20|20|
04119/90054|04619/90054|04089/90054|04854/40054|04416/60054|
EC|1|2|15|1|1|4|4|5453913|616|249|Česká Třebová|1|21|21|1|21|
22|
EC|1|2|15|1|1|4|4|5453613|520|219|Pardubice hl.n.|1|21|55|1|
21|56|05638/80054|00201/10054|
EC|1|1|15|1|1|4|4|5453414|439|220|Kolín|1|22|16|1|22|17|
19221/10054|09480/00054|
EC|1|1|15|1|1|4|4|5457076|326|208|Praha hl.n.|1|22|58|
00375/50054|01774/40054|09165/50054|19015/50054|09964/40054
```

### 4.3.8 Funkcia posledne_upravy()

Niektoré stanice zo vstpných dát majú nekorektné súradnice. Táto funkcia im priradí súradnice ich predchádzajúcej stanice. Takže na mape sa vykreslí len jedna bodka. Okrem toho funkcia vykonáva priradenie do hodnoty zrýchlenia vďaka technickým možnostiam vlaku na 7% jazdy na tomto úseku.

### 4.3.9 Funkcia vlaky_rozdelenie()

Funkcia „vlaky_rozdelenie()“ realizuje vytvorenie index-sekvenčného súboru v adresári „Vlaky“ s jednou úrovňou indexu. Tento adresár bude základným vstupným bodom pre hlavný program simulácie. Postup je veľmi podobný ako v „subor2_rozdelenie()“.

## 4.4 Možné urýchlenia programu

Program by sa dal urýchliť približne o polovicu zmenou reprezentácie sekvenčných súborov KVL_BIN.txt, stanice.txt a subor1.txt na ich index-sekvenčnú formu.

# 5 Simulácia

## 5.1 Užívateľská dokumentácia

### 5.1.1 Inštalácia a spustenie programu

Program vo Windows je možné spustiť cez .exe súbor v adresári „Meškanie vlakov“ na priloženom CD. Pre operačný systém Linux sa v tomto adresári nachádza súbor „Makefile“, pomocou ktorého je možné aplikáciu spustiť.

Program je možné rozbehnúť pod všetkými známejšími operačnými systémami Windows a Linux a pravdepodobne by to šlo aj u Mac Os X. Podrobný návod ako to urobiť je opísaný v angličtine v kapitole 1 a 3 v tutorialy [7] . V skratke ide o to dať do správnych adresárov používaného vývojového prostredia obsah adresárov „lib“ a „include“ z knižníc SDL a patričné .dll súbory dať do adresára, odkiaľ sa spúšta program. Potom je potrebné k programu pridať vstupné dáta: Adresáre Vlaky a Subor2 (4.3.5) a súbory stanice.txt,dni.txt (4.1.2), zoznam_vlakov.txt, mapa-podklad-new.png, arial.ttf, sipky.jpg, pendolino,mpg . Následne už len skompilovať a spustiť program.

### 5.1.2 Ovládanie aplikácie

Celú aplikácia sa dá ovládať klávesnicou, myšou alebo kombináciou obidvoch. V priebehu programu sú vypisujú správy, ktoré oznamujú užívateľovi, čo má urobiť. Sú dvojakého druhu. Pasívne (len výpisy) alebo aktívne, ktoré sú svetlejšie, reagujú na pohyb myšou a keď na ne užívateľ kline, tak sa vykoná nimi opísaná operácia. Tá sa vykoná aj po stlačení príslušnej klávesy alebo tlačidla na obrazovke (šípky vpravo hore alebo klávesnica vľavo vedľa mapy). Užívateľ je

upozorňovaný na chybne zadané vstupy. Uživateľ si najprv zvolí, čí chce generovať štatistiky alebo chce spustiť grafickú simuláciu. V prípade druhej možnosti následno zadá dátum, krokovanie a variantu meškania. Potom si môže zvoliť, či chce pridať číslo vlaku priamo, alebo zadá stanicu zo súboru stanice.txt, odkiaľ má tento vlak odchádzať a program mu ponúkne zoznam vlakov, ktoré z tejto stanice odchádzajú a on si môže jeden vybrať. V prípade zadaného neplatného čísla vlaku je mu ponúknutý zoznam najbližších platných vlakov z ktorých si jeden môže vybrať alebo zadať vlastné číslo. Potom zadá stanicu, do ktorej príde vlak s meškaním a jeho veľkosť. Následne prebehne kontrola, či vlak chodí v daný deň v tej stanici. Po zadaní vstupných dát sa spustí samotná simulácia. V každom jej kroku môže užívateľ pomocou šípky hore pridať do simulácie ľubovoľne veľa vlakov, ktoré idú neskôr ako je aktuálny čas simulácie. Ďalej môže stlačením šípky dole zmeniť krokovanie simulácie. A nakoniec posunúť simuláciu o krok dopredu (šípka vpravo).
Vpravo dole sa  vypisuje zoznam najnovšie zmeškaných vlakov vo formáte:
číslo meškajúceho vlaku | stanica, kam má prísť s meškaním | pravidelný príchod do tejto stanice (hodiny:minúty)| meškanie, s ktorým tam príde (minúty:sekundy).
Podrobnosti o všetkých zmeškaných vlakov sa zapisujú do súboru vystup.txt.
Do súborov „vlaky_uj.txt“ a „vlaky_zm.txt“ sa vypisujú ujdené a zmeškané vlaky počas simulácie. Je tam číslo vlaku a veľkosť meškania.

V prípade generovania štatistík sa tieto súbory volajú  vlaky_var[1-8]_[uj|zm].txt.
1-9 sú varianty meškaní,  „uj“ označuje ujdené vlaky a „zm“ zmeškané vlaky.
Výpočet pri štatistikách prebieha rovnako ako v grafickom prostredí, len bez grafiky. Užívateľ v tomto prípade na začiatku zadá počet vlakov, pre ktoré chce simuláciu previesť a ich meškanie. Potom sa ním určený počet vlakov pre všetky varianty vzájomného meškania bude simulovať. Na konci sa vytvorí súbor suhrn_vysledkov.txt, kde je pre každú variantu meškaní počet ujdených a zmeškaných vlakov a ich priemerné meškanie. Ďalej sú tam vypísane najčastejšie zmeškané (ujdené) vlaky a ich počet, koľkokrát boli zmeškané (ujdené).

## 5.2 Programátorská dokumentácia

### 5.2.1 Vstupné dáta

Hlavným vstupným bodom programu je adresár „Vlaky“ v jeho index-sekvenčnej podobe. Ďalšie vstupné súbory sú:
„dni.txt“ (4.1.2), mapa ČD - „mapa_podklad-new.png“, obrázok so šípkami - „sipky.jpg“, obrázok pendolína - „pendolino.bmp“, font - „arial.ttf“ a „zoznam_vlakov.txt“.
Ak užívateľ nevie čísla vlakov, ktoré chce omeškať, tak je potrebný ešte adresár „Subor2“ (4.3.4-5) a súbor „stanice.txt“ (4.1.5). Slúžia pre ponúknutie vlakov, ktoré odchádzajú z užívateľom zadanej stanice.
Ak má užívateľ podobný alebo rovnaký formát vstupných dát ako som mal ja od ČD, tak ho môže previesť na požadovaný formát pre simuláciu pomocou môjho prevodného programu, ktorí prikladám v adresári Vstupné dáta na CD.

### 5.2.2 Dátové štruktúry programu

#### 5.2.2.1 Objekt Vlak

Najdôležitejšou dátovou štruktúrou v programe je objekt Vlak:

```
class Vlak{
        vector<vector<string> > train;//Tu je celý vlak načítaný zo vstupných dat.
        int location;  //Index pre stanicu, kde sa vlak nachádza v train.
        vector<int> lates; //Je to vektor meškaní vlaku v jednotlivých staniciach.
        Má veľkost train.size(). Je to úmyselne o jedna viac, ako by bolo potrebné (pretože
        prvá stanica v train je až na pozícii 1) kôli zjednodušeniu indexovania.
        int number_days;  //Počet dní od 10.12.2006 do štartovného dňa vlaku. Ak začne
        vlak meškať až v ďalšom dni svojej cesty, aj tak sa udáva štartovný deň vlaku.
        bool change;  //Udáva, či bol vlak zmenený v jednom časovom kroku algoritmu. Ak
        áno, treba ho opäť prerátať. Defaultne je change==true.
```

```
public:
        Vlak();
        Vlak(vector<vector<string> > t,vector<int> ls,int lo,int days,bool );
        void fill(vector<vector<string> > t,vector<int>,int lo,int days,bool);//Vyplnenie položiek
        vlaku
        //V dalších 4 funkciách "r" na začiatku ich názvu označuje return. Vracajú private
        položky triedy Vlak.
        vector<vector<string> > rtrain();//Vráti train
        int rlocation();
        int rlates(int i);
        int rdays();
        bool rchange();

        void increase_lo();//Zvýši pozíciu o 1.
        void decrease_lo();// Zníži pozíciu o 1.
        void change_lates(int i,int v);//Vykoná lates[i]=v.
        void change_x(int i,string s);//Vykoná train[i][c_x]=s.
        void change_y(int i,string s);//Vykoná train[i][c_x]=s.
        void change_number_days(int i);//Zmení poèet dní na hodnotu i.
        void change_change(bool b); //Vykoná change=b.
        void change_location(int i);//Vykoná location=i.
};
```

**Význam jeho dátových položiek:**

- „train“ je vlak z adresára „Vlaky“
- „late“ je aktuálne meškanie vlaku
- „location“ je aktuálna pozícia vlaku
- „mumber_days“ vyjadruje aktuálny dátum vo formáte počtu dní od 10.12.2006

Pre grafický výstup program používa nasledujúce triedy:

Class Timer – Práca s časom. Reguluje počet zobrazení za sekundu.

Class StringInput – Práca so vstupom z klávesnice.

Class Message – Jedna textová správa na obrazovka pripravená ako tlačidlo pre myš.

Class Button – Tlačidla na obrazovke. Používa vektor správ z triedy Message.

Trieda Button slúži pre prácu s tlačidlami na obrazovke. Má dva druhy tlačidiel. Šípky, ktoré sa zobrazujú vpravo hore a správy z triedy Message. Po tom, ako užívateľ vykoná niečo v aplikácii, tak sa prechádza cez jej tlačidlá a vykonáva sa príslušná reakcia.

#### 5.2.2.2 Varianty vzájomného čakanie vlakov podľa priorít

Tri triedy vlakov, 9 variant ich vzájomného čakania pri meškaní. Sú to:

## Varianta 1

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 30 | 30 | 30 |
| R, Ex | 30 | 30 | 30 |
| IC, EC, EN, SC | 30 | 30 | 30 |

Maximálne doby čakania sú u všetkých vlakov zhodné a nastavené na vysokú konštantu bez prihliadania na prioritu. Tento systém sa dlhé roky používal na železniciach.

Pri križovaní je uprednostnený vlak s väčšou prioritou alebo nemeškajúci vlak v prípade rovnosti. Toto pravidlo ostáva rovnaké pre varianty 1-6.

## Varianta 2

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 5 | 5 | 0 |
| R, Ex | 5 | 5 | 0 |
| IC, EC, EN, SC | 5 | 5 | 0 |

Časy meškania sú prevzaté od ČD z roku 2008.

# Varianta 3

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 10 | 5 | 0 |
| R, Ex | 10 | 5 | 0 |
| IC, EC, EN, SC | 10 | 5 | 0 |

Časy meškania sú prevzaté od ČD z roku 2007.

# Varianta 4

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 5 | 5 | 5 |
| R, Ex | 5 | 5 | 5 |
| IC, EC, EN, SC | 5 | 5 | 5 |

Symetrická varianta.

# Varianta 5

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 10 | 5 | 5 |
| R, Ex | 10 | 5 | 5 |
| IC, EC, EN, SC | 10 | 5 | 5 |

Časy meškania sú prevzaté od ŽSR z roku 2008.

# Varianta 6

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 0 | 0 | 0 |
| R, Ex | 0 | 0 | 0 |
| IC, EC, EN, SC | 0 | 0 | 0 |

Vlaky vôbec nebudú na seba čakať s výnimkou križovaní.

# Varianta 7

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 10 | 5 | 5 |
| R, Ex | 10 | 5 | 5 |
| IC, EC, EN, SC | 10 | 5 | 5 |

Časy zostávajú rovnaké ako vo variante 5, ale menia sa pravidlá pri križovaní protismerných vlakov. Je tu snaha priblížiť sa k realite, ako sa rozhodujú výpravcovia. Avšak vzhľadom k neúplnosti vstupných dát to nie je úplne možné. Zmena spočíva v tom,  že keď sa križujú vlaky s rovnakou prioritou, tak sa uprednostňuje také križovanie, aby bol po ňom súčet meškaní oboch vlakov čo najmenší. Robí sa to tak bežne v praxi.

# Varianta 8

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 10 | 5 | 5 |
| R, Ex | 10 | 5 | 5 |
| IC, EC, EN, SC | 10 | 5 | 5 |

To isté ako varianta 7, avšak pri križovaní už nie sú uprednostňované vlaky vyššej priority, ale menší súčet ich meškaní po ňom.

# Varianta 9

| | Os, Sp | R, Ex | IC, EC, EN, SC |
|---|---|---|---|
| Os, Sp | 30 | 30 | 30 |
| R, Ex | 30 | 30 | 30 |
| IC, EC, EN, SC | 30 | 30 | 30 |

To isté ako varianta 1, ale pravidlá križovania sú z varianty 8.

### 5.2.3 Implementácia

Úvod k popisu implementácie je v kapitole 3.2. Po tom ako užívateľ zvolí, či chce generovať štatistiky alebo si chce pozrieť simuláciu, sa spustí funkcia statistiky() alebo funkcia simulácia().
Vo funkcii simulacia() užívateľ zadá dátum, časový krok simulácie a variantu meškania. Realizuje to funkcia input_in_simulation(). Následne je vyzvaný na pridanie ľubovoľného počtu vlakov do fronty meškajúcich vlakov v simulácii (vektor v). Pridanie jedného vlaku vrátane jeho prípojov vykonáva funkcia add_next_train(). Tá realizuje aj prvý výpis zadaného vlaku do výstupného súboru „vystup.txt“. Počiatočný čas simulácie je určený užívateľom zadaným dňom, prvým vlakom a stanicou, kam príde neskôr a veľkosťou tohto meškania.
Teraz už sú všetky údaje pre štart simulácie známe a tak sa spustí funkcia solve(). Jej hlavnou časťou je cyklus cez vektor „v“. Každá iterácia tohto cyklu znamená prejdenie jedného časového kroku programu. Vypisujú sa v nej základné údaje tohto kroku do súboru „vystup.txt“. Prebieha tam cyklus cez všetky vlaky vo „v“. Každý z nich má príznak, či už bol v tejto iterácii zmenený (na začiatku iterácie každý, alebo novo pridané prípoje vlakov, ktoré už vo fronte boli ), alebo nie. Pre všetky, ktoré boli zmenené sa spustí funkcia one_train(). Táto funkcia slúži na posúvanie vlaku po jeho dráhe až do času simulácie, zakresľovanie do mapy a výpis do súboru (pomocou funkcie kresli_jeden_usek()), možné vyhodenie z fronty (vlak stratil meškanie alebo prišiel do svojej poslednej stanice) a pridávanie jeho prípojov. Tie však treba ešte v tej istej iterácii prerátavať, lebo je možné, že ich ešte tiež treba posunúť do času v simulácii. Keď už na každý vlak vo fronte v (aj novo pridané) bola aplikovaná funkcia one_train(), iterácia pokračuje a užívateľ môže pridať ďalší zmeškaný vlak, zmeniť čas krokovania, a nakoniec posunúť simuláciu o krok ďalej. To realizuje funkcia input_in_simulation(). Potom sa čas v simulácii posunie o jeden krok a na konci jednej iterácie sa všetkým vlakom priradí príznak 'zmenený' a celé sa

to opakuje až pokiaľ fronta v nie je prázdna.
Vytvorenie fronty prípojov realizuje funkcia solve_one_train(). Rieši hlavne pravidlá križovania vlakov a pre zistenie či daný prípoj má skutočne meškať (čakacie doby) volá funkciu time(). Spojenie fronty meškajúcich prípojov s frontou „v“ vykonáva funkcia join().
Nasleduje podrobnejší popis vybraných funkcií v približne chronologickom poradí ako nasledujú za sebou v programe

#### 5.2.3.1 Funckia simulation()

Slúži na zadanie počiatočných vstupných dát simulácie ako je dátumu, krokovanie a varianta meškania. Potom zavolá funkciu add_next_train(), ktorá pridáva užívateľom pridané vlaky, vrátene ich prípojov, do fronty pomocou funkcie join(). Nakoniec zavolá funkciu solve(), ktorá dostane frontu meškajúcich vlakov a vyrieši celú simuláciu.

#### 5.2.3.2 Funkcia add_next_train()

Na vstupe dostane frontu meškajúcich vlakov a počiatočné údaje simulácie. Funkcia vyzve užívateľa na zadanie nového vlaku (číslo vlaku, stanica, meškanie) a pridá ho do fronty meškajúcich vlakov. Položku “number_days” - štartovací deň jazdy vlaku nastaví podľa pozície stanice, do ktorej má vlak prísť s meškaním a času (čas v simulácii v minútach od 10.12.2006). Teda vlak, ktorý má trasu na viac dní, nemusí ísť zo svojej začiatočnej stanice v aktuálny deň simulácie. Kontroluje sa, či chodí v dni vyrátanom z čísla “number_day” a tiež či príde do zadávanej stanice pred časom simulácie. O toto sa stará funkcia check_date_of_new_train(). Funkcia pridá do fronty “v” len vlaky, ktoré sú aktuálne za časom simulácie. Simulovať minulosť nemá význam. Následne sa spustí pre tento vlak vo fronte funkcia solve_one_train(), ktorá nájde jeho prípoje a funkcia join() ich pripojí k vektoru “v”, ak splňujú podmienky v nej obsiahnuté.

#### 5.2.3.3 Funkcia solve_one_train()

Na vstupe dostane vlak (bude označený ako vlak č.1. Prípojový bude vlak č.2) , ktorý príde s meškaním do nejakej stanice svojej cesty. Ak je to potrebné (napr. vlak musí čakať na prioritnejší vlak) zvýši jeho meškanie. Ak je to práve práve užívateľom zadaný vlak, tak meškanie sa nemení. Pretože vlak toto meškanie mohol získať napr. haváriou medzi dvoma stanicami a tak už v tej prvej nemôže čakať na prioritnejší vlak.

Funkcia pridá do fronty „v1“ všetky vlaky, ktoré budú skutočne čakať na meškajúci vlak vrátane ich meškaní v nasledujúcej stanici ich cesty. Teda po tej, do ktorej má prísť vlak 1 s meškaním a kde oni na neho musia čakať.

Funkcia funguje nasledovne. Zoberie všetky prípoje vlaku 1 v udanej stanici. Zistí, či chodia v daný deň. Ak áno, tak ich rozdelí na tie, ktoré idú oproti vlaku 1 (fronta „op“, ale len pri jednokoľajových tratiach) a tie ostatné (fronta pp). S frontou op funkcia ďalej pracuje. Podľa zadanej varianty sa volí čo najvhodnejšie križovanie protismerných vlakov. Tu môže nastať ešte zvýšenie čakacej doby vlaku 1.

Pri variantách 8 a 9 funkcia vyberá také križovanie, aby súčet meškaní oboch vlakov bol čo najmenší. Robí to tak, že porovná, ktorý vlak by viac čakal na ten druhý pri oboch križovaniach. Ak je uprednostnený vlak č.2, tak ho vyhodí z „op“ a ak je to potrebné (vlak č.1 musí čakať na príchod vlaku č.2), tak zvýši meškanie vlaku 1.

Pri ostatných variantách dáva vždy prednosť vlaku s vyššou prioritou, pričom zvyšuje podľa potreby meškanie vlaku č.1. Ak ma vyššiu prioritu vlak č.2, tak ho vyhadzuje z „op“. Ráta aj so zrýchlením protismerného vlaku vďaka jeho technickým možnostiam.

Pri rovnosti priorít pri variante 7 funkcia uprednostňuje variantu križovania podľa nižšieho súčtu meškaní. Pri ostatných variantách (1-6) má prednosť vlak č.1.

Pre zostávajúce vlaky z „op“ a všetky z „pp“ sa spustí funkcia time().

#### 5.2.3.4 Funkcia time()

Funkcia spočíta, či vlak vo fronte má skutočne meškať. Je to v prípade, ak rozdiel príchodu (s meškaním) predchádzajúceho vlaku a jeho odchodu je kladný a zároveň nie je väčší, ako povoľuje užívateľom zadaná varianta časov meškania jednotlivých typov vlakov (1-7). Kontroluje aj prechod cez polnoc. Ak sú splnené všetky podmienky, tak funkcia zistí správny čas meškania a priradí vlak do „v1“.

#### 5.2.3.5 Funkcia join()

Funkcia pridáva ďalšie vlaky do fronty meškajúcich vlakov. Pri tom môžu nastať 4. prípady, ktoré sú už popísané vrátane ich riešení v kapitole 3.2.1.

#### 5.2.3.6 Funkcia solve()

Popísaná v úvode do implementácie v kapitole 5.2.3

#### 5.2.3.7 Funkcia one_train()

Funkcia rieši jazdu jedného vlaku až do času v simulácii vrátane pridávania jeho prípojov, vykresľovania na mapu a výpis do výstupných súborov. Pozícia v tomto vlaku je nastavená na stanicu, kde má vlak prísť s meškaním. Najprv sa skontroluje, či vlak príde to tejto stanice vrátane jeho meškania ešte pred časom simulácie. Ak áno, tak sa vlak vykreslí na mapu pomocou funkcie kresli_jeden_usek(), posunie sa o jednu pozíciu a vypočíta sa mu nové meškanie (funkcie decrease_late() a solve_one_train()). Celé sa to opakuje až pokiaľ vlak nepríde do konečnej stanice, nestratí sa meškanie alebo ďalší úsek jazdy vlaku sa už nemôže vykresliť, pretože do tej stanice dorazí neskôr ako je čas simulácie.

#### 5.2.3.8 Funkcia input_in_simulation()

Spolu s funkciou zadaj_string() sa starajú o grafický výstup v programe. Najprv vymaže staré výpisy, vypíše aktuálne a potom spracováva vstupy z klávesnice a myši od uživateľa za pomoci tried StringInput, Button a Timer.

# 6 Vyhodnotenie výsledkov

## 6.1 Tabuľky

Všetky tabuľky sú pre 350 postupne zmeškaných vlakov.

Pri meškaní 25 minút pre všetky vlaky je tabuľka nasledovná:

| 25m | # uj | Meškanie | # | % | # zm | Meškanie | # | % | #zm+ # uj |
|---|---|---|---|---|---|---|---|---|---|
| 1 | >10000 | 0 | 0 | 0,000 | >400000 | 0 | 0 | 0,000 | 0 |
| 2 | 1241 | 21 | 12 | 0,378 | 581 | 9 | 8 | 0,250 | 1822 |
| 3 | 1378 | 22 | 13 | 0,388 | 1687 | 10 | 10 | 0,552 | 3065 |
| 4 | 1212 | 22 | 12 | 0,370 | 615 | 9 | 8 | 0,270 | 1827 |
| 5 | 1328 | 22 | 13 | 0,374 | 1805 | 10 | 10 | 0,574 | 3133 |
| 6 | 1245 | 19 | 16 | 0,274 | 92 | 14 | 2 | 0,111 | 1337 |
| 7 | 947 | 19 | 12 | 0,279 | 3100 | 8 | 14 | 0,636 | 4047 |
| 8 | 941 | 19 | 12 | 0,277 | 2913 | 8 | 12 | 0,662 | 3854 |
| 9 | 85 | 43 | 3 | 0,028 | 161217 | 10 | 641 | 0,671 | 161302 |

Tabuľka 5.1

Pri meškaní 15 minút to vyzerá nasledovne:

| 15m | # uj | Meškanie | # | % | # zm | Meškanie | # | % | #zm+ # uj |
|---|---|---|---|---|---|---|---|---|---|
| 1 | >10000 | 0 | 0 | 0,000 | >400000 | 0 | 0 | 0,000 | 0 |
| 2 | 779 | 21 | 8 | 0,356 | 592 | 9 | 6 | 0,361 | 1371 |
| 3 | 1163 | 25 | 8 | 0,501 | 2055 | 10 | 9 | 0,645 | 3218 |
| 4 | 752 | 22 | 8 | 0,343 | 632 | 9 | 6 | 0,385 | 1384 |
| 5 | 1104 | 26 | 8 | 0,483 | 2199 | 10 | 10 | 0,634 | 3303 |
| 6 | 768 | 14 | 10 | 0,270 | 47 | 10 | 1 | 0,116 | 815 |
| 7 | 463 | 15 | 8 | 0,175 | 4300 | 8 | 19 | 0,642 | 4763 |
| 8 | 446 | 15 | 6 | 0,258 | 3995 | 8 | 16 | 0,669 | 4441 |
| 9 | 12 | 39 | 2 | 0,000 | 61807 | 10 | 306 | 0,609 | 61819 |

Tabuľka 5.2

A posledná tabuľka je pri meškaní 5 minút.

| 5m | # uj | Meškanie | # | % | # zm | Meškanie | # | % | #zm+ # uj |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 19652 | 48 | 283 | 0,234 | 310436 | 15 | 1349 | 0,647 | 330088 |
| 2 | 234 | 26 | 4 | 0,178 | 298 | 8 | 4 | 0,259 | 532 |
| 3 | 208 | 28 | 4 | 0,143 | 445 | 9 | 6 | 0,257 | 653 |
| 4 | 228 | 26 | 4 | 0,170 | 305 | 8 | 4 | 0,267 | 533 |
| 5 | 298 | 29 | 4 | 0,259 | 461 | 9 | 6 | 0,270 | 759 |
| 6 | 167 | 10 | 4 | 0,089 | 17 | 5 | 1 | 0,002 | 184 |
| 7 | 12 | 15 | 2 | 0,000 | 405 | 6 | 3 | 0,475 | 417 |
| 8 | 12 | 15 | 1 | 0,000 | 384 | 6 | 3 | 0,456 | 396 |
| 9 | 0 | 0 | 0 | 0,000 | 463 | 7 | 4 | 0,420 | 463 |

Tabuľka 5.3

Vysvetlivky k tabuľkám:

1. stĺpec: Číslo varianty- nastavení pravidiel meškania vlakov
2. stĺpec: Počet ujdených vlakov
3. stĺpec: Priemerné meškanie (keby neušli) ujdených vlakov
4. stĺpec: Počet najčastejšie ujdeného(ých) vlaku(ov) [slabé miesto]
5. stĺpec: Pravdepodobnosť, že keď sa náhodná simulácia spustí stokrát, tak medzi ujdenými vlakmi bude(ú) aj ten(tie) z predchádzajúceho stĺpca
6. stĺpec: Počet zmeškaných vlakov
7. stĺpec: Priemerné meškanie zmeškaných vlakov
8. stĺpec: Počet najčastejšie zmeškaného(ých) vlaku(ov) [slabé miesto]
9. stĺpec: Pravdepodobnosť, že keď sa náhodná simulácia spustí stokrát, tak medzi zmeškanými vlakmi bude(ú) aj ten(tie) z predchádzajúceho stĺpca
10. stĺpec: Súčet zmeškaných a ujdených vlakov

## 6.2 Analýza tabuliek

### 6.2.1 Tabuľka 5.1

Prvé, čo určite udrie do očí sú veľmi nepriaznivé výsledky z varianty 1. Keďže táto varianta sa používala dlhé roky na železnici, tak nemôže byť až taká zlá, ako ju ukazujú výsledky zo simulácie. Ich dôvodom je to, ako sú určené prípoje vo vstupných dátach (4.3.6) . Dá sa predpokladať, že väčšina z nich sú protismerné vlaky na jednokoľajových tratiach a to spôsobuje tak nepriaznivé výsledky.
Túto hypotézu potvrdzujú aj výsledky z deviatej varianty, ktorá má rovnaké čakacie doby ako prvá, ale už funguje so zmenou križovaní tak, aby súčet meškaní bol čo najmenší. Táto varianta dopadla podstatne lepšie ako prvá, i keď z dôvodu vyšších časov meškania pochopiteľne horšie ako zvyšné varianty 2-8. Varianta 1 je tiež vysoko výpočetne náročná.

Varianty 2,4 a 3,5 sú si veľmi podobné. Pravdepodobne preto, že tabuľky sa líšia len v časoch prípustného meškania vlakov najvyššej priority, ktorých je pomerne málo. Miernym prekvapením je, že varianty 2, 4 dopadli mierne lepšie nielen v počte zmeškaných vlakov (tam je to výrazné), ale aj čo sa týka počtu ujdených vlakov. Je to spôsobené veľkým počtom zmeškaných vlakov, ktoré spôsobujú ujdenie iných vlakov. Súčasné vstupné dáta ukazujú na zistenie, že je najefektívnejšie, aby vlaky na seba čakali čo najmenej. Potvrdzujú to aj výsledky z varianty 6, ktorej súčet zmeškaných a ujdených vlakov dopadol výrazne lepšie ako v predchádzajúcich variantách.

Výsledky varianty 7 sú veľmi zaujímavé. Ukazuje, že rozkladanie meškania medzi viaceré vlaky rovnakej priority je správny postup na zníženie počtu ujdených vlakov.

Tých je výrazne menej v porovnaní s predchádzajúcimi variantami s rovnakými čakacími dobami. Nevýhodou je o dosť väčší počet zmeškaných vlakov.

Varianta 8 mierne na prvý pohľad prekvapuje v tom, že je tam len veľmi malý rozdiel oproti variante 7. Je to spôsobené tým, že program za protismernú trať označuje len tú, kde sa zhodujú obe stanice dvoch vlakov. U jedného je to predchádzajúca a u druhého následná. Toto sa však u vlakov s rôznou prioritou príliš často nestáva.

### 6.2.2 Tabuľky 5.2 a 5.3

Obe tabuľky sa vo veľkej miere zhodujú z výsledkami z tabuľky 5.1, pomer jednotlivých položiek je približne rovnaký, len počty sa zmenili. Nastalo však aj pár rozdielov. Prvým je zväčšenie rozdielov vo výsledkoch pri 15 minútovom meškaní. Je to však pochopiteľné, pretože väčšina čakacích dôb je do 15 minút a tak tu sa aj najviac prejaví rôznosť jednotlivých variant.
Druhým rozdielom je výrazne zlepšenie varianty 9 pri 5 minútovom meškaní. Je to preto, že pri algoritme výberu menšieho súčtu dvoch meškajúcich vlakov sa má meškanie skôr tendenciu strácať ako prehlbovať. Napr. tým, že v tomto prípade až dva vlaky môžu využiť svoje technické možnosti na zníženie toho istého meškania, ktoré by inak musel znižovať jeden vlak. Tiež tým, že pri rozložení meškania môže prísť aj k jeho skráteniu.

### 6.2.3 Slabé miesta v grafikone

Vypočítam pravdepodobnosti zmeškania alebo ujdenia jedného konkrétneho vlaku po 100 simuláciách náhodne zvoleného zmeškaného vlaku. Pri variante 1 je pravdepodobnosť zmeškania najčastejšie meškaného vlaku #/#zm (označenia z tabuliek). Pri 100 vlakoch bude daný vlak zmeškaný s pravdepodobnosťou 1- ( (#zm-#)/#zm)^100. Hodnota tohto čísla je uvedená v tabuľkách v piatom a deviatom stĺpci. S pozorovania vidíme, že čím je vyšší počet vlakov, tak tým je aj vyššie toto číslo. Teda ak je nejaký vlak slabým miestom na železnici, tak so zhoršujúcou sa variantov sa jeho zlé zaradenie do grafikonu prejavuje viac.
V praxi je dôležité najčastejšie meškané alebo ujdené vlaky poznať a hľadať príčiny, prečo to tak je a snažiť sa ich odstrániť.

## 6.3 Príčiny skreslenia výsledkov zo simulácií

V praxi sa zoznam možných prípojov určuje v každom grafikone ako samostatný dokument. Avšak ten nemám. Takže ako prípoje sú označené všetky vlaky, ktorých odchod zo stanice prichádzajúceho vlaku je menší ako „n“, kde „n“ je konštanta 30 minút.
Je to nedokonalé riešenie, pretože tu nie sú úplne vyriešené vlaky, ktoré idú v protismere. Protismerný vlak je taký, ktorý ide do stanice, z ktorej ma prísť prichádzajúci vlak. Avšak ma to problém.
Dajme tomu, že máme tri stanice: A,B,C. Os ide A->B->C. Ec ide C->A. Meškanie 20 minút. Program vyhodnotí, že EC nemá čakať na príchod Os a pravdepodobne dôjde ku fiktívnej zrážke. Avšak pretože nemám v dátach ani zoznam skutočných prípojov, ani zoznam trati, tak tento problém sa nedá riešiť. Dalo by sa to vylepšiť

tak, že program bude kontrolovať cestu nie len jednu stanicu dopredu, ale v tomto prípade tu by tu bol problém s možnými cyklami na trati.

Taktiež v programe bez zoznamu trati nemôžem vyriešiť obiehanie dvoch vlakov idúcich rovnakým smerom, kedy rýchlejší, v niektorej stanici, kde nestojí, obieha pomalší, ktorý tam stojí.

## 6.4 Skvalitnenie výsledkov zo simulácií

Z analýzy tabuliek vidíme, že výsledky zo štatistík sú pomerne výrazne ovplyvnené nedostatočnými vstupnými dátami. Avšak ak by si užívateľ zaobstaral kvalitnejšie vstupné dáta, hlavne zoznam prípojov pre jednotlivé vlaky, a previedol by ich do formátu požadovaného simulačným programom, tak by aj výsledky zo simulácie boli kvalitnejšie, bližšie k realite.

# 7. Záver

Cieľom tohto projektu bola grafická simulácia meškania vlakov pri rozličných vstupných časoch čakacích dôb, vytváranie štatistík a hľadanie slabých miest v grafikone. Všetky tieto úlohy program spĺňa. Bola zvolená efektívna reprezentácia vstupných dát pre vyhľadávanie v nich (index - sekvenčný súbor) .

Nad rámec zadania boli vypracované tri varianty aj s praktickými zmenami pri križovaní vlakov. Simuláciou bolo dokázané, že tieto pravidlá pomerne efektívne napomáhajú znižovaniu počtu ujdených vlakov v sieti.

Aplikácia je spustiteľná pod platformami Windows a Linux bez úpravy zdrojového kódu. Objektový návrh sprehľadňuje kód a uľahčuje jeho opätovné použitie. Je navrhnutá tak, aby ju bolo veľmi ľahké rozširovať o nove funkcionality.

## 7.1 Možné rozšírenia a využitia aplikácie

Program by bolo možné priblížiť viac realite dodaním všetkých potrebných vstupných dát a následným menším prispôsobením programu.
Veľmi zaujímavá možnosť je, že by program v budúcnosti mohol pomáhať pri rozhodovaní výpravcov a vlakových dispečerov ohľadom meškania vlakov.
V tom prípade by bolo treba vylepšiť grafické rozhranie hlavne o manuálnu možnosť rozhodovania pri každom vlaku a tiež program by mal vedieť nielen prevádzať simuláciu dopredu, ale aj krokovanie dozadu po jednotlivých krokoch.
Program je využiteľný aj pre iné druhy dopravy. Stačí mu dať požadované vstupné data. V staršej verzii už bol použitý aj pre simuláciu v MHD Košice. Prikladám pre zaujímavosť na CD v adresároch Kosice1 a Kosice2.

# Literatúra

Nasledujúce zdroje som používal pre vytvorenie tejto práce a ich autorom týmto ďakujem.


[1] ČESKÉ DRÁHY, a.s. – Generální ředitelství
Odbor osobní dopravy a přepravy: *Čekací doby a opatření při zpoždení vlaků osobní dopravy,* 2007-2008.

[2] Železničná spoločnosť Slovensko, a.s.
Sekcia marketingu: *Čakacie časy a opatrenia pri meškaní vlakov osobnej dopravy,* 2007-2008.

[3] Generálne riaditeľstvo ŽSR
Odbor rozvoja generálneho riaditeľstva ŽSR: *Dopravný predpis Ž1,* 2005

[4] *Bruce Eckel*: Myslíme v jazyku C++ (Grada 2000)

[5] *Bruce Eckel, Chuck Allison*: Myslíme v jazyku C++ 2.díl (Grada 2006)

[6] Knižnice SDL (http://www.libsdl.org/)

[7] Tutorial SDL (http://lazyfoo.net/SDL_tutorials/)

[8] Tutorial SDL (http://www.root.cz/serialy/sdl-hry-nejen-pro-linux/)

[9] Vývojové prostredie Mingw (http://www.mingw.org/)

[10] Index sekvenčný súbor (http://mrr.wz.cz/is/dokumentace.htm)