Škoda Auto Vysoká škola o.p.s.

Zápis a čtení dat mezi programem LINGO a textovými soubory

Název předmětu

Datum zpracování

Jakub Procházka

Obsah

1 Úvod

V rámci optimalizačních modelů a simulací se program LINGO často využívá k analýze a řešení komplexních problémů v oblasti logistiky, dopravy a dalších inženýrských disciplín. Jednou z klíčových vlastností tohoto nástroje je jeho schopnost efektivně pracovat s externími datovými soubory, což umožňuje přizpůsobit modely specifickým požadavkům uživatele. Tento proces zahrnuje nejen načítání dat z externích souborů, ale i jejich exportování do přehledného formátu, který je snadno čitelný pro lidského uživatele.

Textové soubory jsou v operačním systému macOS často preferovaným formátem pro práci s daty v programu LINGO, zejména kvůli omezením při používání funkce OLE (Object Linking and Embedding), která je na této platformě pro práci s Excel soubory nedostupná. Na rozdíl od Windows, kde lze OLE efektivně využívat pro přímé propojení s Excel soubory, macOS neumožňuje tuto formu integrace. Proto se pro načítání a zápis dat často používají právě textové soubory (TXT), které jsou jednoduše formátovatelné a umožňují bezproblémovou výměnu dat mezi LINGO a externími aplikacemi. Tento přístup zajišťuje vysokou míru flexibility a kompatibility s různými platformami, což je výhodné při práci na macOS

Čtení a zápis dat jsou nezbytné pro efektivní propojení programů s reálnými daty, ať už jde o optimalizaci přepravy, analýzu nákladů nebo simulace dodavatelských řetězců. Tento proces je realizován pomocí několika různých metod, které jsou vhodné pro specifické typy datových souborů. V první části kapitoly se zaměříme na funkci @READRM, která umožňuje načítat data z textových souborů. V druhé části se zaměříme na funkci @FILE, která poskytuje alternativní způsob načítání dat.

Dále budeme diskutovat, jak exportovat výsledky z LINGA do přehledných textových souborů pomocí příkazů jako @TEXT, @WRITE a @TABLE, které umožňují zápis dat do souborů a jejich správnou strukturalizaci. Výsledné soubory mohou být následně využity pro další analýzu nebo pro vizualizaci dat v přehledném formátu.

2 Načítání dat z textového souboru do LINGA

Načítání dat z textového souboru je klíčovým krokem v mnoha optimalizačních modelech, kde jsou externí data důležitá pro analýzu a rozhodovací procesy. V programu LINGO je možné načítat data z textových souborů pomocí různých funkcí, které umožňují flexibilní a efektivní import dat do modelu. Tento proces je nezbytný pro zajištění správného propojení mezi externími datovými soubory a modelem, zejména při práci s dynamickými a nestrukturovanými daty.

V této podkapitole se zaměříme na různé metody načítání dat ze souborů .txt. První metodou je využití funkce @READRM, která umožňuje načítání dat po jednotlivých záznamech, a to až do určeného oddělovače. Druhou metodou je funkce @READLN, která čte data po jednotlivých řádcích. Třetí metodou je funkce @FILE, která umožňuje načítat více datových hodnot současně z TXT dokumentu do modelu.

2.1 Načítání dat pomocí funkcí @READRM/@READLN

Načítání dat z externích souborů, zejména textových souborů (.txt), je nezbytným krokem při práci s optimalizačními modely v programu LINGO, kde jsou externí data důležitá pro analýzu a rozhodovací procesy.

Tyto dvě funkce umožňují efektivní import dat z textových souborů, přičemž každá z nich má svůj specifický způsob použití. Funkce @READRM je navržena pro načítání dat po jednotlivých záznamech, které jsou oddělené specifikovaným oddělovačem (např. středníkem, tabulátorem nebo jiným znakem). Tento způsob čtení je ideální pro soubory, kde jsou hodnoty odděleny určitým znakem a je potřeba tato data načíst do jednotlivých proměnných. Naopak funkce @READLN načítá data po jednotlivých řádcích, což je vhodné pro soubory, kde je každá hodnota nebo záznam na samostatném řádku.

2.1.1 Funkce @READRM

Funkce @READRM (Read Record by Marker) je používána pro načítání dat z textového souboru, přičemž umožňuje definovat specifické oddělovače, kterými mohou být různé znaky, jako například středník, čárka nebo tabulátor. To znamená, že uživatel má plnou kontrolu nad tím, jakým způsobem budou jednotlivé položky odděleny a načítány.

Tato flexibilita je velmi užitečná při práci s různými formáty datových souborů, kde se struktura může lišit.

Příklad použití funkce @READRM pro načítání dat

V následujícím příkladu využíváme funkci @READRM k načítání dat z textového souboru. Data jsou následně ukládána do množin a parametrů, jako jsou DODAVATEL, ODBERATEL, a, b a c. Funkce @READRM umožňuje načítání hodnot na základě specifikovaného oddělovače, v tomto případě středníku (;), čímž zajišťuje správné rozdělení hodnot do odpovídajících proměnných. Tento příklad ukazuje, jak efektivně načítat a organizovat data z textového souboru v modelu LINGO.

Příklad kódu:

```
CALC:
    DODAVATEL = @READRM('Dopravni problem1.txt', ';');
    ODBERATEL = @READRM( , ';');
    a = @READRM( , ';');
    b = @READRM( , ';');
    c = @READRM( , ';');
ENDCALC
```

1. **Načítání množiny DODAVATEL**

   Řádek *DODAVATEL = @READRM('Dopravni problem1.txt', ';');* načítá hodnoty do množiny DODAVATEL. Funkce @READRM otevře soubor Dopravní problém1.txt a začne načítat data až do prvního výskytu oddělovače, kterým je středník (;). Tento krok je zásadní pro správné oddělení hodnot a jejich následné přiřazení do požadovaných proměnných.

2. **Načítání množiny ODBERATEL**

   Na řádku *ODBERATEL = @READRM(, ';');* již není nutné znovu specifikovat název souboru, protože LINGO si tento název pamatuje z předchozího volání funkce. Funkce pokračuje v načítání dat ze souboru *Dopravni problem1.txt* a načítá hodnoty do množiny ODBERATEL, opět až do středníku (;).

3. **Načítání parametrů a, b, c**

   Načítání hodnot parametrů a, b a c probíhá podobným způsobem. Funkce @READRM načítá hodnoty postupně z jednoho souboru, přičemž pro každý parametr

specifikuje místo, kde načítání končí (opět středník). Tento přístup umožňuje efektivně číst různé sekce dat bez potřeby opakovaného zadávání názvů souboru nebo parametrů.

Tento příklad ukazuje, jak lze pomocí funkce @READRM jednoduše načítat data z textového souboru a přiřazovat je do proměnných, což je velmi užitečné pro práci s externími datovými soubory v modelech LINGO. Funkce @READRM je flexibilní a efektivní způsob, jak získat data oddělená specifikovanými znaky, a zároveň minimalizuje potřebu opakovaného zadávání názvů souborů, což zjednodušuje a urychluje práci s daty.

Struktura datového souboru

Soubor *Dopravni problem1.txt* by měl mít následující strukturu, kde každý blok dat je ukončen středníkem (;), viz Obr. 1

```
Plzen
CeskeBudejovice
Jihlava;

Tabor
Pribram
JinHradec
Pisek;

110 160 180;

90 130 80 120;

10      8       20      9
9       13      6       13
7       11      10      18;
```

Obr. 1 Vstupní soubor pro funkci @READRM

2.1.2 Funkce @READLN

Funkce @READLN (Read Line) je v programu LINGO určena pro čtení celých řádků textu z externích souborů. Na rozdíl od funkce @READRM, která umožňuje načítání dat po jednotlivých položkách oddělených specifikovaným znakem (například středníkem nebo tabulátorem), funkce @READLN čte celý řádek textu jako jednu hodnotu.

Využití funkce @READLN

Funkce @READLN je ideální pro různé scénáře, kde je potřeba jednoduše načíst celé řádky textu bez nutnosti detailního dělení dat. Prvním scénářem je jednoduchý přístup k datům, kdy soubor obsahuje celkový textový blok nebo nestrukturované informace, které není třeba dále členit. Pokud data v souboru nejsou rozdělena do jednotlivých položek, ale spíše představují kompletní textové řádky, @READLN nabízí efektivní řešení pro načítání těchto informací bez nutnosti se zabývat jejich podrobným formátováním.

Dalším typickým využitím funkce je načítání celých řádků, což je užitečné, pokud každý řádek souboru obsahuje samostatnou informaci, kterou chceme načíst jako jednu hodnotu. Příkladem mohou být soubory s komentáři, logy nebo jinými textovými záznamy, kde je každý řádek autonomní jednotkou. V těchto případech @READLN umožňuje efektivní načtení celého řádku včetně všech znaků a mezer, což zjednodušuje práci s nestrukturovanými daty.

Díky své jednoduché implementaci je funkce @READLN také velmi snadná na použití, protože nevyžaduje žádné definování oddělovačů mezi hodnotami, jak je tomu u funkce @READRM. Stačí pouze specifikovat soubor, a funkce načte celý řádek jako jeden celek. Tento přístup značně zjednodušuje kód a činí proces načítání dat efektivním, zejména při práci s méně strukturovanými nebo volně formátovanými daty.

Hlavní rozdíl mezi funkcemi @READRM a @READLN spočívá v tom, jakým způsobem čtou data. Funkce @READRM je určena pro načítání dat po jednotlivých položkách, kde jsou hodnoty odděleny specifikovaným znakem (např. středníkem nebo tabulátorem). Tato funkce je ideální pro strukturované soubory, které obsahují data ve formátu, který lze snadno rozdělit na jednotlivé položky.

Obě tyto funkce umožňují flexibilitu při práci s různými formáty souborů a dynamicky reagovat na jejich strukturu. V závislosti na tom, jak je soubor strukturován, si uživatel může vybrat funkci, která mu umožní efektivně načíst požadované hodnoty. Funkce @READRM a @READLN musí být zapsána v sekci CALC, která je nepostradatelná pro práci s dynamickými daty. Sekci DATA, jako je tomu u funkce @OLE použít v tomto případě nelze.

2.1.3 Použití sekce CALC místo sekce DATA

Sekce CALC v LINGO je primárně určena pro provádění výpočtů, ale také poskytuje široké možnosti manipulace s daty, což je klíčové při práci s externími datovými zdroji, jako jsou textové soubory nebo PDF dokumenty. Tyto zdroje často obsahují dynamické a různorodé struktury, které není možné přesně definovat předem. Například soubory mohou mít různé formáty, počty řádků, oddělovače a další formátovací prvky, které je třeba správně interpretovat pro úspěšné načtení dat.

V tomto kontextu nabízí sekce CALC výhodu oproti sekci DATA, která je určena pro práci s pevně stanovenými datovými strukturami. Zatímco sekce DATA je limitována definováním konkrétních datových struktur, CALC umožňuje dynamické načítání dat z externích souborů. To je zvláště užitečné při práci s textovými soubory, jejichž struktura se může lišit, a umožňuje uživatelům reagovat na aktuální formát souboru a přizpůsobit kód podle potřeb modelu.

Sekce CALC poskytuje flexibilitu i v oblasti manipulace s načítanými daty. Pomocí funkcí, jako je @READRM, @FILE a dalších, mohou uživatelé specifikovat způsob, jakým budou data načítána, včetně definování znaků nebo oddělovačů, jako jsou středníky, tabulátory nebo vlnovky. Tato schopnost umožňuje efektivní zpracování souborů s různými formáty a flexibilní úpravu načítání dat podle aktuálních požadavků.

Další výhodou sekce CALC je možnost pokročilého řízení manipulace s daty během jejich načítání a exportu. Funkce, jako je @READRM, nabízejí nejen možnost definovat oddělovače, ale i specifikovat přesnou strukturu čtení dat, což je nezbytné pro práci s různými typy souborů. Tento dynamický přístup poskytuje flexibilitu, která není dostupná v sekci DATA.

2.2 Načítání dat pomocí funkce @FILE

V této kapitole se zaměříme na alternativní způsob načítání dat do programu LINGO pomocí funkce @FILE. Tato funkce slouží k přímému importu dat z textového souboru do sekce DATA a na rozdíl od funkce @READRM zde není potřeba využívat sekci CALC.

Funkce @FILE je navržena pro situace, kdy je soubor organizován v pevně definovaných blocích, kde každý blok dat končí specifickým oddělovačem, v tomto případě

vlnovkou (~). Tento způsob načítání je vhodný pro soubory, které nevyžadují flexibilní čtení po řádcích nebo dynamické přizpůsobení dat, jak je tomu u funkcí @READRM a @READLN.

Použití funkce @FILE je nejvhodnější, pokud máte soubor s dobře definovanou strukturou, kde jsou data rozdělena do bloků pomocí vlnovky. V následujícím příkladu je načítání dat realizováno prostřednictvím funkce @FILE, která zajišťuje načtení množin a parametrů přímo v sekci DATA.

Příklad použití funkce @FILE pro načítání dat

V následujícím příkladu ukážeme, jak využít funkci @FILE k načítání dat z textového souboru. Data se načítají přímo do proměnných a parametrů, které jsou uvedeny před symbolem -". Funkce @FILE je vhodná pro soubory, které mají pevně danou strukturu, kde bloky dat jsou odděleny vlnovkou (~). Tento příklad demonstruje, jak efektivně načítat a organizovat data z externího souboru v modelu LINGO.

Příklad kódu:

```
DATA
    DODAVATEL, a = @FILE('Dopravni problem2.txt');
    ODBERATEL, b = @FILE('Dopravni problem2.txt');
    c = @FILE('Dopravni problem2.txt');
ENDDATA
```

1. Načítání množiny DODAVATEL a parametru a
   Řádek *DODAVATEL, a = @FILE('Dopravni problem2.txt');* načítá data do množiny DODAVATEL a parametru a. Funkce @FILE otevře soubor Dopravni problem2.txt a načte hodnoty až do vlnovky ( ), která označuje konec bloku dat. Tento krok je klíčový pro správné přiřazení názvů dodavatelů do množiny DODAVATEL a kapacit dodavatelů do parametru a.

2. Načítání množiny ODBERATEL a parametru b
   Další řádek *ODBERATEL, b = @FILE('Dopravni problem2.txt');* načítá data do množiny ODBERATEL a parametru b. Funkce @FILE opět pokračuje ve čtení ze stejného souboru, přičemž načítá hodnoty až do vlnovky ( ), která odděluje jednotlivé bloky dat. Tento krok umožňuje přiřadit názvy odběratelů do množiny ODBERATEL a požadavky odběratelů do parametru b.

3. Načítání parametru c

   Poslední řádek *c = @FILE('Dopravni problem2.txt');* načítá hodnoty do parametru c, který reprezentuje náklady na přepravu. V tomto případě funkce @FILE načte data až do konce souboru, protože poslední sekce dat není oddělena žádným specifickým znakem (vlnovkou) a je tedy považována za konec datového souboru.

Struktura datového souboru

Soubor Dopravni problem2.txt by měl mít následující strukturu, kde každý blok dat je ukončen vlnovkou (~), viz Obr. 2

```
Plzen 110
CeskeBudejovice 160
Jihlava 180 ~

Tabor 90
Pribram 130
JinHradec 80
Pisek 120 ~

10      8       20      9
9       13      6       13
7       11      10      18
```

Obr. 2 Vstupní soubor pro funkci @FILE

Tato struktura umožňuje přehledné členění dat, kde každá skupina (např. dodavatelé, odběratelé, náklady na přepravu) je jasně oddělena koncovou vlnovkou. Funkce @FILE automaticky identifikuje konec každého bloku a přiřazuje hodnoty do příslušných proměnných a parametrů v LINGU.

Shrnutí

Použití sekce CALC a funkcí @READRM a @READLN umožňuje flexibilní načítání dat z externích souborů v programu LINGO. Funkce @READRM je vhodná pro načítání dat oddělených specifickými oddělovači, zatímco @READLN umožňuje číst celé řádky textu bez nutnosti definování oddělovačů. Tyto funkce jsou ideální pro práci s nestrukturovanými soubory, kde se data mohou měnit podle potřeby. Naopak funkce @FILE, používaná v sekci DATA, umožňuje načítat strukturované soubory s definovanými oddělovači a přímo přiřazovat hodnoty do parametrů. Kombinace těchto metod

zajišťuje efektivní manipulaci s různými formáty dat v LINGO na platformě macOS, kde není možné načítat data přímo z tabulek z MS Excel.

3 Export dat z LINGA do textového souboru

V této kapitole se budeme věnovat procesu exportu výsledků z modelu v LINGO do externího textového souboru. Exportování dat je klíčovým krokem pro dokumentaci a analýzu výsledků modelování, protože umožňuje uživatelům snadno přenášet a sdílet výstupy mimo samotné prostředí LINGO. Pro tento účel se v LINGO používají funkce @TEXT, @WRITE a @TABLE, které umožňují strukturovaně zapisovat data a výsledky do textového souboru.

Každá z těchto funkcí má specifický účel a umožňuje přizpůsobit výstupní soubor dle požadovaného formátu. Funkce @TEXT a @WRITE jsou používány k zápisu dat, zatímco funkce @TABLE je určena pro export tabulkových dat a jejich organizované zapsání, což je zvláště užitečné při exportu matic nebo rozsáhlých výsledků.

Tento přístup umožňuje snadno přizpůsobit strukturu výstupu, což je užitečné pro další analýzu nebo reportování výsledků modelu v externích aplikacích. V následující části se podíváme na konkrétní příklady použití těchto funkcí a ukážeme si, jak efektivně exportovat data z LINGO do textového souboru.

3.1 Export dat pomocí funkce @TEXT

Funkce @TEXT v LINGO slouží k zapisování textového obsahu do výstupního souboru, což je užitečné, když je potřeba přidat popisy, vysvětlivky nebo oddělit části výstupu nadpisy. Funkce je obvykle kombinována s ostatními příkazy pro export dat, jako jsou @WRITE a @TABLE, čímž se dosahuje přehledné a strukturované prezentace výsledků.

Příklad použití funkce @Text pro export dat

Příklad kódu pro export dat do textového souboru:

```
{DATA:
  @TEXT('Vystup.txt') = @WRITE('OBJEM PREPRAVY',@NEWLINE(1));
  @TEXT('Vystup.txt') = @WRITE(40*'-',@NEWLINE(1));
  @TEXT('Vystup.txt') = @TABLE(x);
  @TEXT('Vystup.txt') = @WRITE(@NEWLINE(1));
  @TEXT('Vystup.txt') = @WRITE('UCELOVA FUNKCE',@NEWLINE(1));
  @TEXT('Vystup.txt') = @WRITE(40*'-',@NEWLINE(1));
  @TEXT('Vystup.txt') = @WRITE('    ', z);
ENDDATA}
```

Export dat v LINGU umožňuje přehledné ukládání výsledků z optimalizačního modelu do externího textového souboru pro další analýzu. Použití funkcí jako @TEXT, @WRITE, @TABLE a @NEWLINE usnadňuje vytváření dobře strukturovaných a čitelných výstupů. Níže je vysvětleno, jak každá z těchto funkcí pracuje a jak přispívá k organizaci exportovaného souboru.

1. **@TEXT**

   Příkaz @TEXT zahajuje zápis do specifikovaného souboru. Například @TEXT ('Vystup.txt') určuje, že všechny následující výstupy budou zapisovány do souboru *Vystup.txt*. Tento soubor je vytvořen nebo otevřen pro zápis v okamžiku, kdy je tento příkaz proveden. Použití @TEXT je užitečné, pokud potřebujeme ukládat výsledky nebo jiné textové informace přímo do souboru, což umožňuje snadný přístup k výsledkům po ukončení výpočtu.

2. **@WRITE**

   Funkce @WRITE je určena pro zápis textu nebo hodnot přímo do výstupního souboru. Například příkaz:

   ```
   @TEXT('Vystup.txt') = @WRITE('OBJEM PREPRAVY',
      @NEWLINE(1));
   ```

   zapíše text "OBJEM PREPRAVY"do výstupního souboru *Vystup.txt* a následně přidá právě jeden nový řádek pomocí funkce @NEWLINE(1). Díky tomu je možné do souboru vkládat konkrétní výsledky nebo názvy sekcí, což zajišťuje srozumitelnost výstupu.

3. **@TABLE**

   @TABLE se používá pro export strukturovaných tabulkových dat. Pokud model generuje více hodnot, například objemy přepravy, @TABLE umožňuje zapisovat tyto hodnoty do tabulky. Například:

   ```
   @TEXT('Vystup.txt') = @TABLE(x);
   ```

   zapíše do výstupního souboru tabulku, ve které budou uvedeny hodnoty proměnné x představující objemy přepravy mezi různými lokacemi. Tento přístup je výhodný zejména pro výstupy s velkým množstvím dat, kde je potřeba přehledné a konzistentní uspořádání výsledků.

4. **@NEWLINE**

   Příkaz @NEWLINE(1) přidává do výstupního souboru prázdný řádek, což přispívá k lepší čitelnosti exportovaných dat. Například po zápisu hodnot pomocí @WRITE může @NEWLINE(1) sloužit k ukončení aktuálního řádku a přesunu kurzoru na nový řádek. Opakovaným použitím tohoto příkazu lze také vytvářet mezery mezi různými sekcemi, což přehledně odděluje jednotlivé části výstupu. Hodnota v závorce u příkazu udává počet řádků, které chceme přidat.

5. **Oddělení sekcí**

   Pro lepší orientaci ve výstupním souboru může být vhodné vizuálně oddělit jednotlivé části výsledků. K tomu slouží příkaz:

   ```
   @TEXT('Vystup.txt') = @WRITE(40*'-', @NEWLINE(1));
   ```

   Tento řádek vloží do souboru horizontální čáru složenou ze 40 pomlček, což vytváří jasné vizuální oddělení mezi různými sekcemi. Oddělovací čáry jsou zvláště užitečné při práci s rozsáhlými daty, kde zvyšují čitelnost výstupu a usnadňují identifikaci klíčových částí.

6. **Zápis hodnoty účelové funkce**

   Na závěr je zapsána hodnota účelové funkce, která představuje optimalizační výsledek modelu. Například:

   ```
   @TEXT('Vystup.txt') = @WRITE('    ', z);
   ```

   Tento příkaz zapíše do výstupního souboru několik mezer a poté hodnotu z, což je výsledná hodnota účelové funkce. Tento zápis poskytuje přehled o efektivitě modelu a umožňuje rychlou interpretaci optimalizačního výpočtu.

Příklad exportovaného výstupu

Kombinací všech uvedených funkcí je možné vytvořit strukturovaný výstupní soubor, který přehledně shrnuje všechny klíčové výsledky modelu a poskytuje uživateli snadno čitelný přehled pro další analýzu. Výstupní soubor by v tomto konkrétním případě vypadal například takto, viz Obr. 3

```
OBJEM PREPRAVY
----------------------------------------
                     TABOR    PRIBRAM  JINHRADEC      PISEK
 PLZEN            0.000000   40.00000   0.000000   70.00000
 CESKEBUDEJOVICE  0.000000   0.000000   80.00000   50.00000
 JIHLAVA          90.00000   90.00000   0.000000   0.000000

UCELOVA FUNKCE
----------------------------------------
 3700
```

Obr. 3 Vstupní soubor pro funkci @FILE

Shrnutí

Tato kapitola poskytuje ucelený návod na export dat a výsledků z modelu v LINGU do externího textového souboru. Exportní kód v této kapitole umožňuje uživateli snadno přenášet výsledky modelu do přehledně formátovaného souboru, který lze dále využít pro analýzu nebo dokumentaci. Použitím kombinace příkazů pro zápis textu, hodnot a tabulek s jasným oddělením jednotlivých sekcí vytváříme strukturovaný výstup, který přispívá k lepší čitelnosti a organizaci dat.