Programovací techniky: Práce s kolejištěm v Laboratoři řízení kolejových vozidel

Modelové kolejiště v laboratoři je elektronický počítačem řízený technologický celek, který s okolím komunikuje (mimo jiné) přes webové rozhraní. Programy, kterými lze kolejiště ovládat, tedy musí obvykle určitým dotazem ze serveru získat informace o okamžitém stavu příslušného prvku a následně sestavit povel, který se opět pošle na server kolejiště. Typy příkladů, které můžeme řešit v laboratoři, můžeme rozdělit do několika obtížnostních úrovní:

1. Program pošle do kolejiště povel (například „výhybka 150 -> do odbočky“).

2. Program zjistí stav prvku a na jeho základě vyšle povel se změněným stavem (například „výhybka 150 -> změna polohy“)

3. Program vyhledá příslušný prvek podle názvu nebo umístění, pak zjistí jeho stav a na základě získané informace vyšle povel (například „přehoď výhybku č. 7 ve stanici Okrouhlá“).

Moduly

Pro řešení úloh jsou připraveny moduly, jejichž operace umožňují získávat informace do interní paměťové struktury, prohledávat je a posílat povely na server kolejiště.

Tři moduly, které jsou potřeba, mají názvy `mptvlakyo`, `mzjisti` a `mtypy`.

Potřebné datové typy:

`UkPolozka` – ukazatel na dynamickou paměťovou strukturu, v níž jsou umístěna získaná data. Její podrobný popis není pro řešení úloh potřebný.

Druhým potřebným typem je VarZaznam, k němuž je k dispozici následující:

```
  TypUdaj = (typretez, typcele, typrealne, typlogik, typpole,
typobjekt);
  VarZaznam = record
    case Udaj: TypUdaj of
     typretez:  (retez: string);
     typcele:   (cele: word);
     typrealne: (realne: single);
     typlogik:  (logik: boolean);
     typpole:   (pole: UkHodnota);
     typobjekt: (objekt: UkPolozka);
  end;
```

Tento typ uchovává informace získané z položek popisujících jednotlivé prvky kolejiště. Opět pro řešení úloh vynecháme nepodstatné popisy typů `UkHodnota` a `UkPolozka`. Vše ostatní jsou jednoduché datové typy.

Dostupné operace:

1. `procedure ZiskejData(BinSoubor, T: string; AdrId: word);`

Uloží data získaná z kolejiště do speciálního souboru, jehož název je v prvním parametru. Parametr `T` udává typ požadovaných dat a může nabývat těchto hodnot:

a) `'bloky'` – získání informací o všech technických prvcích celého kolejiště. V tomto případě se `AdrId` nastavuje na nulu.

b) `'blokStav'` – získání detailu o technickém prvku se zadanou adresou `AdrId`.

c) `'loks'` – získání informací o všech hnacích vozidlech zaznamenaných na serveru (nemusí být fyzicky na kolejišti). `AdrId` se zde nastavuje na nulu.

d) `'lokStav'` -- získání detailu o hnacím vozidle se zadanou adresou `AdrId`.

2. `procedure PosliData(TypPrvku: string; AdrId: word; Polozka: string; Data: VarZaznam);`

Pošle povel do kolejiště, jde o ovládání konkrétního prvku. `TypPrvku` může nabývat hodnot `'blokStav'` nebo `'lokStav'` (ovládáme technický prvek nebo hnací vozidlo), `AdrId` je adresa prvku, `Polozka` je název položky, jejíž hodnotu chceme změnit, a parametr `Data` obsahuje hodnotu, kterou chceme do této položky nastavit. Datový typ hodnoty záleží na typu položky (viz `VarZaznam`).

3. `procedure ZpracujSoubor(BinSoubor: string; var Hlav: UkPolozka);`

Převede speciální soubor získaný procedurou `ZiskejData`, jehož jméno je v prvním parametru, do paměťové struktury, na niž ukazuje ukazatel `Hlav`.

4. `procedure Vyhledej(H: UkPolozka; HlPol, HlHod, VysPol: string; var VysHod: VarZaznam);`

Vyhledání informací v paměťové struktuře, na niž ukazuje parametr `H`. Procedura pracuje ve dvou režimech:

a) Vyhledání cílové hodnoty podle hodnoty jiné položky. Chceme například získat adresu lokomotivy, jejíž název je „Hektor“. Napřed tedy najdeme záznam, v němž položka „nazev“ má hodnotu „Hektor“ a pak z tohoto záznamu vezmeme data v položce „adresa“:

```
Vyhledej(H, 'nazev', 'Hektor', 'adresa', Data);
```

V proměnné `Data` se objeví hodnota adresy záznamu o lokomotivě s názvem Hektor.

b) Pouhé vyhledání hodnoty jedné položky. Použijeme tehdy, má-li struktura pouze jeden záznam, tedy v okamžiku, kdy máme načten detail jednoho prvku. V tomto případě je třetí parametr prázdný řetězec a hodnota druhého a čtvrtého parametru je shodná. Například chceme v detailu lokomotivy zjistit hodnotu položky „smer“:

```
Vyhledej(H, 'smer', '', 'smer', Data);
```

Nástin řešení jednotlivých úloh:

V úlohách kategorie 1:

a) zjištění vstupního údaje z příkazového řádku nebo z proměnné prostředí (použití `ParamCount`, `ParamStr` nebo `GetEnvironmentVariable`)

b) sestavení povelu

potřebujeme například nastavit rychlostní stupeň lokomotivy na 10, což je celočíselná hodnota. Je deklarováno `Data: VarZaznam;`, následně můžeme nastavit:

```
Data.Udaj:=TypCele;
Data.cele:=10;
```

c) použití `PosliData`

```
PosliData('lokStav', Adresa, 'rychlostStupne', Data);
```

V úlohách kategorie 2:

a) zjištění vstupních údajů (stejné jako u 1)

b) získání detailu příslušného prvku. Příklad: chceme zjistit polohu výhybky s adresou `Adresa`:

```
ZiskejData('mujsoubor', 'blokStav', Adresa);
ZpracujSoubor('mujsoubor', Pamet);
        // ukazatel Pamet ukazuje na strukturu v paměti
```

c) získání hodnoty příslušné položky

```
Vyhledej(Pamet, 'poloha', '', 'poloha', Data);
       // v Data typu VarZaznam máme výsledek, poloha je typu
typretez
```

d) zpracování a úprava hodnoty této položky

Je-li v `Data.retez` poloha `'+'`, změníme na `'-'` a opačně

e) sestavení povelu a použití PosliData

```
PosliData('blokStav', Adresa, 'poloha', Data)
```

V úlohách kategorie 3:

Příklad: přehoďte výhybku s označením „B V1“

a) zjištění vstupních údajů (stejné jako u 1 a 2); v proměnné `Oznaceni` budeme mít například „B V1“.

b) získání informací o všech prvcích daného typu (bloky nebo lokomotivy)

```
ZiskejData('mujsoubor', 'bloky', 0);
ZpracujSoubor('mujsoubor', Pamet);
```

c) vyhledání identifikace prvku podle jiné položky (název apod.)

```
Vyhledej(H, 'nazev', Oznaceni, 'id', Data);
       // v proměnné Data získáme celočíselnou adresu výhybky
Adresa:=Data.cele;
```

d) získání detailu prvku podle identifikace (dále je to stejné jako u kategorie

```
ZiskejData('mujsoubor', 'blokStav', Adresa);
ZpracujSoubor('mujsoubor', Pamet);
       // ukazatel Pamet ukazuje na strukturu v paměti
```

e) získání hodnoty příslušné položky

```
Vyhledej(Pamet, 'poloha', '', 'poloha', Data);
       // v Data typu VarZaznam máme výsledek, poloha je typu
typretez
```

f) zpracování a úprava hodnoty této položky

Je-li v `Data.retez` poloha `'+'`, změníme na `'-'` a opačně

g) sestavení povelu a použití `PosliData`

```
PosliData('blokStav', Adresa, 'poloha', Data)
```

Názvy a možné hodnoty použitých položek jsou uvedeny u zadání jednotlivých úloh.