

**TECHNICKÁ UNIVERZITA V LIBERCI**

Fakulta strojní

**KÓDOVÁNÍ VSTUPNÍCH DAT PRO INFORMAČNÍ  
SYSTÉMY ČERPACÍCH STANIC**

Libor Adamec

Diplomová práce

1999

# TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra aplikované kybernetiky

Obor 23 - 40 - 8

Automatizované systémy řízení ve strojírenství

zaměření :

Automatizace inženýrských prací

## KÓDOVÁNÍ VSTUPNÍCH DAT PRO INFORMAČNÍ SYSTÉMY ČERPACÍCH STANIC

Libor Adamec

Vedoucí diplomové práce : Prof. Ing. Vladimír Věchet, CSc.

Technická univerzita Liberec

Konzultant :

Ing. Michal Moučka

Technická univerzita Liberec

### Rozsah práce a příloh :

Počet stran textu : 44

Počet obrázků : 20

Počet tabulek : 2

Počet grafů : 0

Počet příloh : 4





## **ZADÁNÍ DIPLOMOVÉ PRÁCE**

Jméno a příjmení

**Libor A d a m e c**

obor

**23 – 40 - 8    Automatizované systémy řízení ve  
strojírenství**

zaměření

**automatizace inženýrských prací**

Ve smyslu zákona č. 111/1998 Sb. o vysokých školách se Vám určuje diplomová práce na téma:

### **Kódování vstupních dat pro informační systémy čerpacích stanic**

#### **Zásady pro vypracování:**

(uveďte hlavní cíle diplomové práce a doporučené metody pro vypracování)

1. Seznamte se s informačním systémem používaným pro kalibrované nádrže u čerpacích stanic pohonných hmot.
2. Určete strukturu vstupních dat pro používaný informační systém a způsob jejich zakódování
3. Sestavte aplikaci na rozhraní Win32 API pro generování zakódovaných vstupních dat informačních systémů
4. Ověřte funkčnost sestavené aplikace při předání vstupních dat informačnímu systému

V 104/99 S

+ DISKET

KKY/ASŘ (AIP)

43 A., 1. vyd. příl. - DISKET

Forma zpracování diplomové práce:

- průvodní zpráva cca 30 stran + přílohy
- grafické práce dle potřeby

Seznam literatury (uveďte doporučenou odbornou literaturu):

MOUČKA, M. : Sledovací programový systém kalibrovaných nádrží pro čerpací stanice. DP SF – 395, Liberec, TU 1998.

SIMON, R.J. – GOUKER, M. – BARNES, B.C. :Win32 API – průvodce vývojáře. Brno, UNIS PUBLISHING 1997.

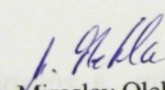
RICHTER, J. : Windows pro pokročilé a experty. Praha, Computer Press 1997.

Vedoucí diplomové práce:

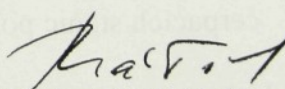
Prof. Ing. Vladimír Věchet, CSc.

Konzultant diplomové práce:

Ing. Michal Moučka

  
Prof. Ing. Miroslav Olehla, CSc.  
vedoucí katedry



  
Doc. Ing. Ludvík Prášil, CSc.  
děkan

V Liberci dne 3. 3. 1999

Platnost zadání diplomové práce je 15 měsíců od výše uvedeného data. Termíny odevzdání diplomové práce jsou určeny pro každý studijní rok a jsou uvedeny v harmonogramu výuky.



TÉMA :

## **KÓDOVÁNÍ VSTUPNÍCH DAT PRO INFORMAČNÍ SYSTÉMY ČERPACÍCH STANIC**

ANOTACE :

Práce se zabývá informačním systémem pro sledování objemu skladovaných materiálů v kalibrovaných nádržích. Její součástí je vytvoření programového produktu na kódování vstupních dat pro informační systém.

SUBJECT :

## **INPUT DATA ENCODING FOR INFORMATION SYSTEMS OF PETROL STATIONS**

ANNOTATION :

This work deals with information system for observing bulk of materials stored in gauged tanks. Its part is programme product for input data encoding for information system.

**Prohlášení :**

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.

V Licerci, dne : 24. 5. 1999

Libor Adamec

*Libor Adamec*



## Poděkování :

Chtěl bych poděkovat Prof. Ing. Vladimíru Věchetovi, CSc,  
i ing. Michalu Moučkovi za pomoc, rady a připomínky.

V Liberci, dne : 27. 5. 1999

Libor Adamec

**OBSAH :**

|                                                                                                 |           |
|-------------------------------------------------------------------------------------------------|-----------|
| <b>1. ÚVOD DO PROBLEMATIKY KALIBROVANÝCH NÁDRŽÍ</b>                                             | <b>6</b>  |
| <b>2. PROGRAMOVÝ PRODUKT - INFORMAČNÍ SYSTÉM EXCALIBER</b>                                      | <b>7</b>  |
| 2.1. Charakteristika programu                                                                   | 7         |
| 2.2. Způsob načtení vstupních dat programem                                                     | 7         |
| <b>3. VSTUPNÍ DATA PRO INFORMAČNÍ SYSTÉM</b>                                                    | <b>9</b>  |
| 3.1. Struktura vstupních dat                                                                    | 9         |
| 3.2. Organizace souboru s daty                                                                  | 13        |
| 3.3. Algoritmus zašifrování souboru                                                             | 14        |
| 3.4. Převod mezi kódovými stránkami CP 1250 a CP 852                                            | 15        |
| <b>4. VÝVOJ APLIKACÍ NA ROZHRANÍ WINDOWS API</b>                                                | <b>16</b> |
| 4.1. Úvod do aplikačního programového rozhraní Windows (Win API)                                | 16        |
| 4.1.1. Win 3.x API                                                                              | 17        |
| 4.1.2. Win 32 API, implementace ve Win32s, Win95, WinNT                                         | 18        |
| 4.2. Mechanismus fungování aplikací v prostředí Windows                                         | 19        |
| 4.2.1. Okno, třída okna                                                                         | 19        |
| 4.2.2. Zpráva, struktura zprávy, zasílání a zpracovávání zpráv                                  | 20        |
| 4.2.3. Realizační funkce okna                                                                   | 21        |
| 4.3. Vývoj aplikace                                                                             | 23        |
| 4.3.1. Vývojové prostředí, postup při sestavení aplikace                                        | 23        |
| 4.3.2. soubor *.c                                                                               | 25        |
| 4.3.3. soubor *.rc                                                                              | 27        |
| 4.3.4. soubor *.def                                                                             | 28        |
| 4.3.4. soubor *.h                                                                               | 29        |
| <b>5. PROGRAMOVÉ PRODUKTY KODER, VSTUP</b>                                                      | <b>30</b> |
| 5.1 Popis aplikace Koder                                                                        | 30        |
| 5.3 Popis modulu Vstup                                                                          | 41        |
| <b>6. ZÁVĚR</b>                                                                                 | <b>42</b> |
| <b>Seznam použité literatury</b>                                                                | <b>44</b> |
| <b>Přílohy</b>                                                                                  |           |
| Příloha č.1 – výpis zdrojového textu programu Koder                                             | 45        |
| Příloha č.2 – výpis zdrojového textu modulu Vstup                                               | 69        |
| Příloha č.3 - tabulky znaků nejpoužívanějších kódových stránek                                  | 72        |
| Příloha č.4 - disketa se zdrojovými texty, sestavenou aplikací KODER, přeloženým modulem VSTUP. |           |



## 1. ÚVOD DO PROBLEMATIKY KALIBROVANÝCH NÁDRŽÍ

Pro skladování pohonných hmot se používá několik typů (tvarů) nádrží.

K nejpoužívanějším patří :

- Válcové ležaté
- Válcové stojaté
- Krabicovité

Častým požadavkem při provozu nádrže je zjištění okamžitého objemu média v nádrži. Ten však přímo změřit nelze. Používá se tzv. kalibrační tabulka nádrže, což je vyjádření objemu v závislosti na výšce hladiny, kterou změřit lze.

Kalibrační (litrovací) tabulky jsou uživatelům dodávány kalibračními středisky, například akreditovaným střediskem kalibrace nádrží akciové společnosti Geodézie ČS se sídlem v Liberci.

Pro sledování okamžitého objemu médií v nádržích slouží programový produkt Excaliber, vytvořený ing. Michalem Moučkou. Jeho vstupem je zašifrovaný soubor, obsahující souhrn informací o jednotlivých nádržích.

Součástí této práce je vytvoření programového produktu Koder, jehož funkce spočívá ve shromáždění potřebných informací o nádržích a jejich zašifrování do předávaného souboru.

Další částí této práce je vytvoření dekódovací rutiny, určené k připojení k programovému produktu Excaliber, která zajistí rozšifrování souboru.



## **2. PROGRAMOVÝ PRODUKT - - INFORMAČNÍ SYSTÉM EXCALIBER**

### **2.1. Charakteristika programu**

Program Excaliber je informační systém sloužící k monitorování stavu media v nádržích čerpacích stanic.

Pro jednotlivé typy nádrží zobrazuje graficky zejména :

- tvar nádrže
- kalibrační křivku nádrže
- výšku hladiny media v nádrži

### **2.2. Způsob načtení vstupních dat programem**

Vstupním souborem pro Excaliber je soubor RESOURCE.RES. Obsahuje v nečitelné (záměrně zašifrované) formě údaje o nádržích. Soubor RESOURCE.RES vznikne přejmenováním přípony jednoho z následujících souborů:

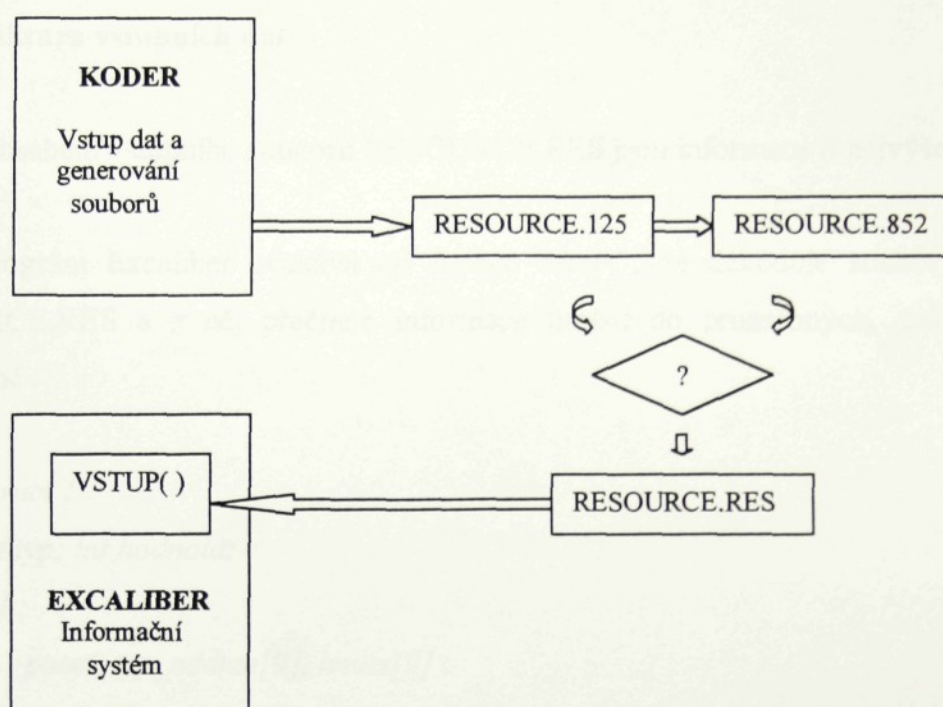
- RESOURCE.125
- RESOURCE.852

To jsou obsahově identické soubory, liší se jen použitou kódovou stránkou znaků. Soubor RESOURCE.125 nese informace ve znakové sadě CodePage 1250 (kódování češtiny Windows CZ), soubor RESOURCE.852 ve znakové sadě CodePage 852 (kódování češtiny Latin II).

Podle toho, s jakou kódovou stránkou prostředí pracuje, uživatel vybere odpovídající soubor.

Soubory RESOURCE.125, případně RESOURCE.852 budou vytvořeny aplikací KODER, která údaje přebírá od obsluhy a zašifruje je.





Obr. 1 Schema spolupráce aplikací

Aplikace KODER je programovým produktem této práce a bude podrobněji popsána v samostatné kapitole.

Program EXCALIBER tedy kromě vlastního jádra musí obsahovat ještě dekódovací část, která zajistí zpřístupnění dat z kódovaného souboru **RESOURCE.RES**. Toto zajistí volání funkce **vstup()**, přidáný k aplikaci Excaliber při jejím sestavení.

Modul **VSTUP** je dalším produktem této práce.

### 3. VSTUPNÍ DATA PRO INFORMAČNÍ SYSTÉM

#### 3.1. Struktura vstupních dat

Obsahem vstupního souboru RESOURCE.RES jsou informace o nejvýše devíti nádržích.

Program Excaliber očekává od funkce *vstup()*, že dekoduje soubor s názvem RESOURCE.RES a z něj přečtené informace umístí do proměnných, deklarovaných následovně :

```
typedef struct {  
    int typ; int hodnota;  
} PLNENI ;  
extern int  pocet, typ_nadrze[9], hmax[9] ;  
extern unsigned char  nazev[9][21], uzivatel[41] ;  
extern float Qmax[9];  
extern float far *Q[9] ;  
extern HGLOBAL objem[9] ;  
extern long barva_produkту[9] ;  
extern PLNENI dole[9], nahore[9] ;  
extern unsigned char ulice[30], mesto[40], telefon[30], mobil[20],  
    fax[20], mail[40], www[30] ;
```

Po zjištění hodnoty hmax (různá velikost kalibrační tabulky pro jednotlivé nádrže) provede funkce *vstup()* alokace a uzamčení paměti pro kalibrační tabulku konkrétní nádrže:

```
dw = (DWORD)( (hmax[i]+1) * sizeof(float) ) ;  
objem[i] = GlobalAlloc ( GPTR , dw ) ;  
if ( objem[i] == NULL ) return (2) ; ;  
Q[i] = (float far *) GlobalLock ( objem[i] ) ;
```

Uvolnění jednotlivých bloků paměti zajistí program Excaliber před svým ukončením.



Tabulka 1. - Podrobný popis jednotlivých proměnných

| Deklarace proměnné:                                                                                | Popis:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>typedef struct {     int typ;     int hodnota; } PLNENI ;  PLNENI dole;  PLNENI nahore;</pre> | <p>Datový typ struktury <i>PLNENI</i> má dva záznamy,<br/> <i>PLNENI.typ</i> je typ plnění<br/> <i>PLNENI.hodnota</i> je hodnota plnění</p> <p>Struktura <i>dole</i> datového typu <i>PLNENI</i>, obsahuje informace o dolní hranici plnění :</p> <p><i>dole.typ</i> :</p> <ul style="list-style-type: none"> <li>1 - 5% plnění</li> <li>2 - výška na konci sání</li> <li>3 - minimální výška</li> </ul> <p><i>dole.hodnota</i> :</p> <p>výška [cm]</p> <p>Struktura <i>nahore</i> datového typu <i>PLNENI</i>, obsahuje informace o horní hranici plnění :</p> <p><i>nahore.typ</i> :</p> <ul style="list-style-type: none"> <li>1 - 95% plnění</li> <li>2 - 97% plnění</li> <li>3 - maximální výška</li> </ul> <p><i>nahore.hodnota</i> :</p> <p>výška [cm]</p> |

|                                    |                                                                                                                                                             |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>int pocet;</i>                  | Počet nádrží                                                                                                                                                |
| <i>int typ_nadrze[9];</i>          | Pole typů jednotlivých nádrží, obsahuje celá čísla :<br><br>0 - pro nádrž válcovou ležatou<br>1 - pro nádrž válcovou stojatou<br>2 - pro nádrž krabicovitou |
| <i>int hmax[9] ;</i>               | Pole maximálních výšek v jednotlivých nádržích, celá čísla, [cm].                                                                                           |
| <i>unsigned char nazev[9][21];</i> | Pole řetězců názvů jednotlivých nádrží, max. 20 znaků pro název nádrže.                                                                                     |
| <i>unsigned char uzivatel[41];</i> | Řetězec jména oprávněného uživatele, max. 40 znaků.                                                                                                         |
| <i>float Qmax[9];</i>              | Pole maximálních objemů v jednotlivé nádrži, reálná čísla.                                                                                                  |
| <i>float far *Q[9];</i>            | Pole pointerů na první hodnotu kalibrační tabulky každé nádrže.                                                                                             |
| <i>HGLOBAL objem[9];</i>           | Pole, obsahující HANDLE paměti alokované pro kalibrační tabulky jednotlivých nádrží.                                                                        |



|                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>long barva_produkту[9];</i>                                                                                                                                                                                                                 | Pole celých čísel obsahujících barvy produktů pro jednotlivé nádrže, číslo odpovídá Windows kódu pro barvu RGB.                                                                                                                           |
| <i>unsigned char ulice[30];</i><br><i>unsigned char mesto[40];</i><br><i>unsigned char telefon[30];</i><br><i>unsigned char mobil[20];</i><br><i>unsigned char fax[20];</i><br><i>unsigned char mail[40];</i><br><i>unsigned char www[30];</i> | Formální údaje o oprávněném uživateli programu:<br><br>Řetězec max. 30 znaků.<br>Řetězec max. 40 znaků.<br>Řetězec max. 30 znaků.<br>Řetězec max. 20 znaků.<br>Řetězec max. 20 znaků.<br>Řetězec max. 40 znaků.<br>Řetězec max. 30 znaků. |

### 3.2. Organizace souboru s daty

Na začátku souboru je úvodní blok, pokračují bloky informací o jednotlivých nádržích, a na konec jsou připojeny formální údaje.

Konkrétní pořadí údajů v souboru :

#### úvodní blok :

- jméno oprávněného uživatele
- počet nádrží

#### blok údajů, týkajících se první nádrže, pořadí údajů :

- název nádrže
- typ nádrže
- kód barvy
- maximální výška
- maximální objem
- úplná kalibrační tabulka (pro rozsah výšek od 0 cm po maximální výšku )
- dolní plnění - typ
- dolní plnění - hodnota
- horní plnění - typ
- horní plnění - hodnota

#### bloky údajů o následujících nádržích

...

#### blok formálních údajů :

- ulice
- město
- telefon
- mobil
- fax
- mail
- www



### 3.3. Algoritmus zašifrování souboru

Pro zajištění ochrany dat i znemožnění neautorizovaného šíření produktu Excaliber, jsou předávané soubory zašifrovány.

#### Návrh algoritmu

První část ASCII tabulky znaků (kódy 0 - 127) je pro všechny znakové sady stejná a obsahuje znaky anglické abecedy, číslice a nejběžnější symboly. Nejspodnější část (kódy 0-31) jsou znaky neviditelné, používané jako řídicí znaky.

Druhá část ASCII tabulky (kódy 128 - 255) obsahuje znaky speciální a je pro různé znakové sady různá.

V případě znakových sad CodePage 1250 (Windows kódování češtiny) a CodePage 852 (MS-DOS kódování češtiny) obsahuje druhá část tabulky kromě symbolů i českou diakritiku. Protože kódováním souboru se nesmí ztratit česká diakritika, je nutné pracovat s celou šířkou 255 znaků ASCII.

Navržený algoritmus je poměrně jednoduchý, přitom ale vyhovující. Pro znaky s kódy 0 - 127 použijí násobení číslem 2. Tak se jejich hodnota, dříve v rozsahu 32 - 127, rozptýlí na 64 - 254. Zároveň se rozptýlí i formátovací znaky (odřádkování a návrat vozíku), takže při otevření šifrovaného souboru textovým editorem nebude viditelná ani podobnost formátu.

Pro znaky s kódy 128-254 použijí součet s číslem 1, a protože je nutné je odlišit od znaků vzniklých násobením, bude jim předcházet znak s ASCII kódem 2, podle něhož později dekodovací rutina pozná, jak ho dešifrovat. Tento znak je jedním z řídicích znaků tisku, při prohlížení textovým editorem ale nepůsobí žádné formátovací změny.

Znak s ASCII kódem 255, (jenž už není kam posunout, neboť kód 256 není k dispozici), bude mít obraz ve znaku s kódem 3.



### 3.4. Převod mezi kódovými stránkami CP 1250 a CP 852

Operační systémy Windows ve verzi CS s českou diakritikou používají pro její reprezentaci kódovou stránku 1250. Jedním z požadavků na aplikaci je, že má být schopná generovat navíc i výstupní soubor, který by ty samé informace obsahoval převedené do kódové stránky 852 tak, aby mohl být využit systémem, používajícím tuto kódovou stránku.

Problém byl vyřešen převodní tabulkou, na základě které bude každý znak Cp1250 nahrazen svým ekvivalentem z CP852. Podkladem pro vznik této převodní tabulky byly kompletní výtisky znakových sad z časopisu Computer Press, viz.příloha č. 3 této práce.

Vytvořená tabulka byla pečlivě ověřena, v aplikaci Poznámkový blok Windows95, vkládáním znaků pomocí ALT xxx pro CP 852, a ALT 0xxx pro CP1250.

(S drženou klávesou levý ALT se na numerické klávesnici vypíše 3-místný ASCII kód znaku ze stránky 852, a po uvolnění ALT je systémem vypsán na místě kurzoru. Pro CP1250 čtyřmístně, s předcházející nulou). Tak je možné vypsát na obrazovku bezprostředně po sobě dva znaky z různých kódových stránek a vidět, zda si odpovídají zcela, či se jen podobají.

Znaky, k nimž nebyl nalezen ani přibližně vypadající protějšek, jsou nahrazeny znakem s kódem 32, tj. mezerou.

V případě existence alespoň přibližně podobného znaku, byl tento použit. Týká se to např. různých typů uvozovek, apostrofů, teček.

Převodní tabulka (přesněji převodní tabulka pro znaky s ASCII kódy 128 - 255) je uložena v pracovním adresáři programu v souboru 1250852.txt a program KODER ji z tohoto souboru přečte. Tím je zároveň dána uživateli možnost změnit převod znaků, u nichž by nesouhlasil s nabízeným přiřazením.

Tabulka je tvořena dvěma sloupci, v prvním je ASCII kód vzoru v sadě 1250, v druhé je ASCII kód obrazu v sadě 852.



## 4. VÝVOJ APLIKACÍ NA ROZHRANÍ WINDOWS API

### 4.1. Úvod do aplikačního programového rozhraní Windows

Aplikační programové rozhraní Windows (Win API, Windows application programming interface) můžeme považovat za soubor nástrojů (funkcí, datových struktur, objektů, souborů ...), které při správném použití umožní vytvořit aplikaci spustitelnou pod Windows.

Prostředí Windows přináší oproti operačnímu systému MS - DOS mnoho rozšíření. Mezi základní vlastnosti systému Windows patří :

- souběžné zpracování úloh - multitasking
- přenos dat mezi aplikacemi
- hardwarově nezávislá grafika
- sdílení hardwaru aplikacemi
- dynamické linkování - knihovny \*.dll

Je zřejmé, že způsob programování aplikací pro platformy Windows bude naprosto odlišný, než programování aplikací pro MS-DOS.

Z programátorského hlediska je nutné přizpůsobit se systému Windows v následujících oblastech :

- Programy musí spolupracovat s Windows způsobem „událostmi řízeného programování“
- Programy musí spolupracovat s grafickým rozhraním (Graphics Device Interface, GDI)
- Programy budou využívat „objekty uživatelského rozhraní“, které jsou k dispozici v rozhraní

Nejčastěji se setkáme s Win3.x API pro 16-bitové verze Windows řady 3.x a Win32 API pro 32-bitové verze Windows. Ve vzhledu aplikací přeložených pro jednotlivé platformy jsou rozdíly nepodstatné, většinu grafických prvků obsahuje už Win 3.x API.



Podstatné rozdíly jsou ale v přístupu k paměti a šířce dat (to vyplývá z toho, že Windows 3.x jsou závislé na MS-DOSu, zatímco Win95 a WinNT jsou zcela samostané operační systémy).

Mluvíme-li o API, je třeba mít na vědomí rozdíl mezi teoretickou definicí rozhraní a konkrétní implementací rozhraní v operačním systému.

#### 4.1.1. Win3.xAPI

Systém Windows 3.x je grafickou nadstavbou 16bitového operačního systému MS-DOS, a tedy pracuje pamětí i daty stejně jako MS-DOS.

Paměť je členěna na segmenty, šířka dat je 16 bitů.

#### 4.1.2. Win32API

Je implementováno v nových operačních systémech, které MS-DOS ke své funkci nepotřebují a přístup k paměti a datům řeší novým způsobem. (vyjímkou je prostředí Win 32 s, bude vysvětleno později).

Šířkou dat je 32 bitů, způsob organizace paměti je od základu přepracován. Paměť není členěna na segmenty, ale adresována lineárně. Aplikace přistupují k paměti jako souvislému bloku bez členění.

Operační systém zajišťuje, aby aplikace směla přistupovat jen ke svému adresovému prostoru a nevznikl konflikt s adresovým prostorem jiné aplikace.

To je zajištěno tím, že aplikaci je předkládán virtuální adresový prostor, a její požadavky na přístup do paměti jsou překládány z virtuálních na fyzické adresy.

(Pokud jde o adresový prostor vlastního operačního systému, ve Windows NT je chráněn proti přístupu ze strany aplikace, a ve Windows 95 chráněn není, ale to už se dostáváme k rozdílům v implementacím).

Další rozdíly Win32API oproti Win3.xAPI jsou např.

- podpora multithreadingu (více běžících „vláken“ v rámci jedné aplikace)
- podpora UNICODE fontů (obsahují v jednom fontovém souboru více národních znakových sad)
- mapování diskových souborů do paměti

a další ..



Win32API z hlediska implementací v operačních systémech :

Rozhraní Win32API je (v současné době) implementováno na čtyřech platformách : Win32s, Windows NT, Windows 95/98 a Windows CE.

Ideálním stavem by bylo, kdyby byly v každé implementaci obsaženy všechny funkce rozhraní Win32, což bohužel není pravda. Aplikaci je nutno vytvářet s ohledem na platformu, na níž se předpokládá její běh.

K jednotlivým platformám :

#### Win32s

Byla vytvořena jako doplnění 16bitových systémů Windows 3.x.

Jejím hlavním úkolem je zajištění konverze 32bitových parametrů funkcí na 16bitové při jejich volání.

Byla vytvořena jako první ze všech, zřejmě proto, aby se podnítil zájem programovat aplikace jako 32bitové. Ostatní platformy byly dostupné až později.

#### Windows NT

Byla v časové ose druhou platformou, podporující rozhraní Win32. Jde o nejvyspělejší operační systém (v rámci produktů firmy Microsoft), je implementován i na jiných procesorech než Intel x86. Existuje ve verzích pro procesory MIPS, Alpha nebo PowerPC.

Obsahuje úplnou implementaci všech funkcí rozhraní Win32.

#### Windows95

Jsou z uživatelského hlediska nástupcem Windows 3.x. Protože v době svého vzniku mířily Windows95 marketingově do oblasti zákazníků chudších, se slabší hardwarovou konfigurací počítače, je implementace Win32 v nich omezena. Nejdůležitějším limitem pro implementaci Win32 na platformě Windows 95 bylo, aby operační systém fungovat už na 4 megabajtech operační paměti.

#### Windows CE

Jsou nejnovější platformou, která podporuje rozhraní Win32. Jsou vyvinuta pro počítače přenosné, až kapesní.

I ony jsou dostupné na neintelovských procesorech.



Pro jaké platformy jsou vytvořeny programy Excaliber a Koder ?

Program Excaliber :

Je aplikace typu Win3.x.

Na platformách Windows 95, Windows NT bude také fungovat, díky tomu, že tyto operační systémy mají vestavěnou část, která se umožňuje běh i 16bitových aplikací.

Bude fungovat i na platformě OS/2 firmy IBM, která je schopna (!!!) provozovat i aplikace pro Microsoft Windows 3.x.

Program Koder :

Je aplikace typu Win32.

Modul Vstup :

Je přeložený samostatně nespustitelný kód, pro Win 3.x.

Důvodem, proč je 16-bitový je to, že bude přilinkován k aplikaci Excaliber, a musí tedy být kompatibilní s jejím 16-bitovým kódem.

## **4.2. Mechanismus fungování aplikací v prostředí Windows**

### **4.2.1. Okno, třída okna**

Okno :

Okno je zásadně důležitý objekt systému Windows. Hlavní okno aplikace, dialogová okna, okna obsahující chybová hlášení. Okny jsou ale i tlačítka, textová pole, editační pole, přepínací či vybírací prvky.

Pro systém Windows je okno prostředkem komunikace. Hlavní okno aplikace pro něj představuje spojení s aplikací.

Okno je identifikováno svým deskriptorem, přiděleným při vytváření. Při vytváření je také spojeno (ukazatelem na jméno funkce) se svou realizační funkcí, která bude vyřizovat zprávy přicházející z tohoto okna.

Ostatní okna aplikace jsou spojena s hlavním oknem aplikace, jsou jím vlastněna. Každé okno zná deskriptor svého rodičovského okna. Jestliže se stane nějaká akce v nějakém okně, Windows pozná, které aplikaci a kterému jejímu oknu akce patří.

Chování dialogových oken může být dvojit.



Okno může být vytvořeno tak, aby se z něj nedalo přepnout do jiného okna (do jiné aplikace však ano). Takové okno nazýváme modálním oknem. Bývá vhodné ho používat tam, kde záleží na tom, aby uživatel v daný čas udělal, co se v okně žádá.

Speciálním případem jsou systémová modální okna pro zobrazení naléhavých zpráv, která nedovolí ani přepnutí do jiné aplikace.

Druhý způsob chování okna je ten, kdy se z něj lze přepnout do jiného okna, okno zůstane v pozadí a uživatel se k němu může vrátit později. Bude vhodné tam, kde nevádí odsunutí požadované činnosti na pozdější dobu.

Činnost v okně je výsledkem spolupráce mezi aplikací a systémem Windows. Windows okno zobrazuje a vykonává některé akce spojené se vzhledem okna, například přemístění nebo změnu velikosti. Aplikace je zodpovědná za vnitřní oblast okna.

#### 4.2.2. Zpráva, struktura zprávy, zasílání a zpracovávání zpráv

Událost, zpráva :

Událostí se rozumí vstup od hardwaru, aplikace nebo systémového časovače. Systém Windows události zachycuje, opatřuje je doplňujícími informacemi a uschovává ve formě tzv. zpráv. V pořadí, v jakém nastaly, jsou uschovány v systémové frontě zpráv a následně rozříděny do jednotlivých aplikačních front zpráv pro aplikace.

Aplikace má ve své hlavní funkci *winmain()* za úkol vybírat zprávy ze své fronty a odesílat je systémovému dipečeru. Ten určí konkrétní okno a zavolá realizační funkci tohoto okna. Tento mechanismus je nazýván mechanismem zpětného volání, anglicky *callback* (odtud pochází deklarace realizačních funkcí slovem *callback*).

Přesně je zpráva struktura, deklarovaná tímto způsobem:

```
typedef struct tagMSG{
    HWND hwnd;
    WORD message;
    WORD wParam;
    LONG lParam;
    DWORD time;
    POINT pt;
}MSG;
```

a jednotlivé proměnné mají následující význam:

- **HWND** - je deskriptor okna, kterému zpráva náleží. Každé okno v systému má jediný deskriptor, tedy je jednoznačně určeno, kterému oknu zpráva patří.



- `message` - obsahuje typ zprávy
- `wParam`, `lParam` - obsahují přídavné, upřesňující informace o zprávě
- `time` - obsahuje čas, při kterém byla zpráva uložena do fronty zpráv
- `pt` - obsahuje souřadnice kurzoru myši v okamžiku uložení zprávy do fronty

Vyzvednutí zprávy z aplikační fronty je přesně zavolání funkce `GetMessage`, která má následující deklaraci :

```
BOOL GetMessage (LPMSG lpMsg, HWND hWnd, WORD wMsgFilterMin, WORD wMsgFilterMax);
```

Funkce vrací hodnotu `TRUE`, jen při vyzvednutí zprávy `WM_QUIT` vrátí `FALSE`.

- `hWnd` - je deskriptor okna, aplikace tedy žádá deskriptorem svého hlavního okna o zprávu, která jí opravdu náleží
- `wMsgFilterMin`, `Max` jsou filtry, umožňující omezit výběr zpráv na zprávy s kódy v určitém rozsahu

#### 4.2.3. Realizační funkce oken

##### Realizační funkce

Realizační funkce okna je funkce, která má reagovat na zprávy, patřící tomuto oknu.

Je to funkce se zpětným voláním, to znamená, že je zavolána systémem Windows (systémovým dispečerem). Jejím vstupem je struktura zprávy.

Realizační funkce může vykonávat nějaké činnosti zavoláním funkcí z API, nebo může sama posílat zprávy. Často se stane obojí najednou, t.j. že zavoláním nějaké funkce dojde zároveň k poslání zprávy.

V případě záměru poslat zprávu existují pro programátora dvě možnosti. Liší se v mechanismu, jakým bude zpráva poslána.

Při použití funkce `PostMessage()` bude zpráva doručena do fronty zpráv, jako všechny ostatní zprávy.

Při použití funkce `SendMessage()` bude zpráva doručena k okamžitému zpracování přímo do smyčky pro zpracování zpráv.



Realizační funkce je deklarována následujícím způsobem :

*LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )*

- hWnd - je deskriptor okna
- uMsg - identifikuje zprávu
- wParam, lParam - obsahují přídatné informace ke zprávě





Legenda ke schématu :

Zdrojové (textové) soubory :

- \*.c - zdrojový program v jazyce C
- \*.h - hlavičkový zdrojový program v jazyce C
- \*.asm - zdrojový program v jazyce assembler
- \*.rc - soubor s definicí prostředků
- \*.def - definiční soubor pro linker

Přeložené soubory :

- objektové soubory na výstupech kompilátorů jazyků assembler a C
- \*.res - zkompilovaný soubor s prostředky
- knihovny připojené linkerem
- \*.exe - sestavená výsledná aplikace

Programy, které se podílí na sestavení aplikace :

- \*.c kompilátor - zkompiluje program v jazyce C, výstupem je \*.obj kód
- \*.asm kompilátor - zkompiluje případné bloky v jazyce symbolických adres, výstupem je \*.obj kód
- \*.rc - zkompiluje zdroje používané aplikací, výstupem je soubor \*.res obsahující všechny zdroje používané aplikací
- linker - sestavuje spustitelný kód modulu, pouze bez přidaného bloku zdrojů

Spojení bloku zdrojů s modulem je až poslední operace. V modulu \*.exe lze najít část se zdroji a používat je odděleně od vlastního spustitelného modulu.

Využívá se toho při spouštění dalších instancí téhož programu, kdy zdroje se zavedou do paměti pouze jednou, pro všechny instance.



#### 4.3.2. Soubor \*.c

Je zdrojovým programem v jazyce C.

Struktura programu:

Zdrojový text programu pro MS-DOS v jazyce C musí obsahovat funkci pojmenovanou *Main()*, a ta je považována za hlavní.

V případě Windows je předepsáno jméno této funkce *Winmain()*.

Tato funkce bude mít za úkol vytvoření hlavního okna aplikace a vybírání zpráv pro aplikaci vybírací smyčkou zpráv.

(Jistě by šlo napsat i aplikaci, která nekomunikuje, nemá okno, a jen tajně cosi dělá, zřejmě to ale nebude častý cíl).

Vytvoření okna znamená zaregistrování jeho třídy, a jeho vykreslení. Při registraci třídy okna má programátor možnost ovlivnit mnoho vlastností.

Registrace prakticky znamená vyplnění struktury *WNDCLASS* informacemi o budoucím okně a zavolání registrační funkce, která třídu zaregistruje.

Deklarace struktury *WNDCLASS*:

```
typedef struct tagWNDCLASS
{
    WORD style;
    LONG *lpfnWndProc;
    int cbClsExtra;
    int cbWndExtra;
    HANDLE hInstance;
    HICON hIcon;
    HCURSOR hCursor;
    HBRUSH hbrBackGround;
    LPSTR lpszMenuName;
    LPSTR lpszClassName;
} WNDCLASS;
```

Význam jednotlivých proměnných :

- style - obsahuje informaci o chování třídy okna. Informace je uložena jako kombinace bitových přepínačů.
- lpfnWndProc - je ukazatel na funkci, která bude zpracovávat zprávy zasílané oknům třídy. Funkce může pracovat pro několik oken současně. Je to dáno tím, že každé okno má svůj deskriptor, podle kterého funkce zjistí, z kterého okna zpráva pochází.



- `cbClsExtra` - počet bajtů přidaných k definici třídy. Slouží k použití v programu. Jsou sdíleny všemi okny třídy.
- `cbWndExtra` - také počet bajtů rezervovaných v datové oblasti. Tato oblast bude ovšem přidělena soukromá pro každé nové okno třídy.
- `hInstance` - deskriptor programu, který třídu registruje
- `hIcon` - deskriptor ikony pro danou třídu
- `hCursor` - identifikátor kurzoru
- `hbrBackGround` - deskriptor štětce, kterým se bude vykreslovat pozadí okna
- `lpzMenuName` - ukazatel na nulou zakončený řetězec identifikující menu okna
- `lpzClassName` - ukazatel na nulou zakončený řetězec jména třídy

Vytvoření okna znamená zavolání vytvářecí funkce *CreateWindow*. Její deklarace :

```
HWND CreateWindow( LPCTSTR lpClassName, LPCTSTR lpWindowName, DWORD  
dwStyle, int x, y, int nWidth, int nHeight, HWND hWndParent, HMENU hMenu, HANDLE  
hInstance, LPVOID lpParam );
```

Jednotlivé parametry určují :

- `lpClassName` - ukazatel na řetězec s jménem třídy, od které bude okno vytvořeno
- `lpWindowName` - ukazatel na řetězec s titulkem okna
- `dwStyle` - styl okna definovaný sadou bitových přepínačů
- `x, y` - pozice okna
- `nWidth, nHeight` - šířka a výška okna
- `hWndParent` - deskriptor rodičovského okna
- `hMenu` - deskriptor menu
- `hInstance` - deskriptor aplikace, vytvářející okno



- `lParam` - pointer na data doprovázející okno

Funkce vrátí `HANDLE` vytvořeného okna.

Dále funkce *Winmain* zajišťuje vybírání zpráv z aplikační fronty.

Funkce *GetMessage* vrací při vybírání vždy `TRUE`, jen při zprávě `WM_QUIT` vrátí `FALSE`. `WM_QUIT` je zpráva oznamující ukončení aplikace a tedy způsobí vyskočení ze smyčky.

Smyčka pro vybírání zpráv bude mít například následující tvar :

```
while (GetMessage( &msg, NULL, 0, 0) )
{
    TranslateMessage( &msg );
    DispatchMessage( &msg );
}
```

Dále program bude obsahovat obsahovat realizační funkci hlavního okna.

V případě, že program vytváří další (dialogová) okna, je třeba zajistit zpracování zpráv od těchto oken. Nejpřehlednější je napsat zvláštní realizační funkci pro každé okno, ale není to podmínkou.

#### 4.3.3. Soubor \*.rc

V API Windows jsou programátorovi k dispozici objekty uživatelského rozhraní. Jsou to grafické prvky s typizovaným vzhledem, které lze ihned použít. Například tlačítka, editační pole, přepínače, rolovací pruhy.

Všechny takovéto objekty jsou z hlediska sestavení aplikace „resources“, neboli „zdroje“.

Informace o „zdrojích“ jsou programátorem ukládány v souboru \*.rc, Při sestavování aplikace je tento soubor kompilován zvláštním programem (resource compiler) a v podobě \*.res je přidán do aplikace.

Prostředky mohou být jednak už zmíněné objekty uživatelského rozhraní, dané k dispozici prostředím, nebo ještě další, vytvářené programátorem.



Jako vytvářené programátorem jmenujme např. bitové mapy s obrázky, ikony, dnes už i multimediální soubory jako zvukové či obrazové úseky.

Každý takový soubor musí být v \*.rc souboru zpřístupněn. Buďto definicí přímo uvnitř souboru \*.rc (např. jednotlivé řady bodů ikony 16 x 16 pixelů), nebo odkazem na jméno externího souboru.

Soubor \*.rc je textový soubor, a je ho možné vytvářet ručně v textovém editoru. Současná vývojová prostředí dávají vývojáři k dispozici editor prostředků, s jehož pomocí jsou soubory \*.rc vygenerovány. Bývá k dispozici i jednoduchý grafický editor na přípravu ikon, případně bitmap.

#### 4.3.4. Soubor \*.def.

Je to definiční soubor pro linker.

Obsahuje informace potřebné k sestavení aplikace.

Některé nejpoužívanější příkazy :

- NAME jméno souboru  
Znamená, že výstupem je přímo spustitelný modul (exe), definuje jeho jméno
- LIBRARY jméno souboru  
Znamená, že výstupem je modul knihovny (dll), definuje jeho jméno
- DESCRIPTION  
Vkládá text do modulu programu, používá se k uložení informace o autorských právech
- CODE  
Definuje atributy kódu v paměti  
MOVEABLE / FIXED - povoluje / zakazuje přesun kódu v případě potřeby operačního systému přeargumentovat paměť  
PRELOAD / LOADONCALL - zavést do paměti předem / zavést až při volání  
DISCARDABLE - umožňuje správci paměti modul uvolnit z paměti v případě jejího nedostatku
- DATA  
Má podobnou funkci jako CODE, týká se paměťového prostoru pro data

MULTIPLE / SINGLE / NONE -

paměť zvlášť pro každou instanci / sdílet paměť instancemi / žádná paměť

- EXPORTS seznam procedur

Připojuje procedury k adresovému prostoru aplikace, bez uvedení procedury v sekci EXPORTS procedura nebude znát globální proměnné

#### 4.3.5. Soubor \*.h

Je hlavičkový souborem, do hlavního \*.c připojen zpravidla na začátku, pomocí direktivy `#include`.

Jeho obsahem bývají definice konstant, prototypy funkcí, dle uvážení programátora. Ve stejném smyslu (k definicím konstant) bývá \*.h soubor využíván moderními editory \*.rc souborů.



## 5. PROGRAMOVÉ PRODUKTY KODER, VSTUP

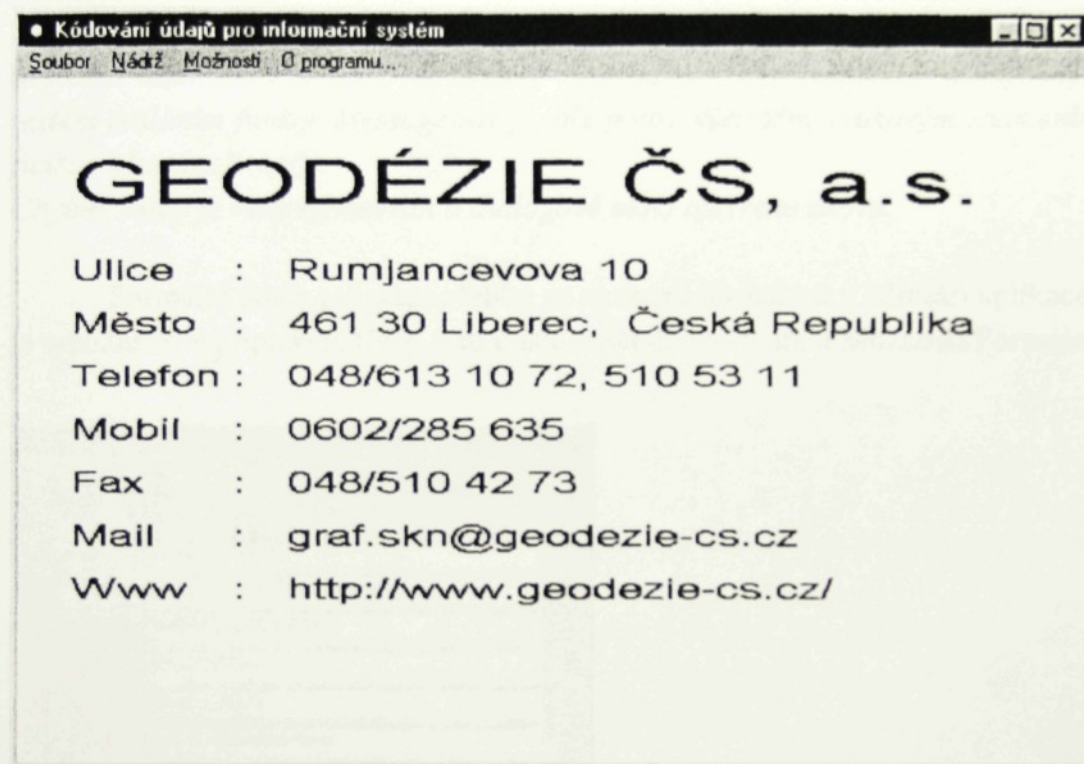
### 5.1. Popis aplikace Koder

(Legenda :

Běžným písmem je aplikace charakterizována z hlediska uživatele.

Proložené písmo je užito pro přiblížení přístupu programátora).

Po spuštění aplikace se zobrazí hlavní okno programu :



Obr. 3. Hlavní okno aplikace Koder, úvodní informace

Okno obsahuje titulkový pruh s titulkem programu a systémovými ikonami a menu pro výběr akcí. Jako úvodní obsah okna jsou zobrazeny formální informace o Středisku kalibračních nádrží.

*Třída tohoto okna je zaregistrována a okno je vytvořeno v hlavním bloku WinMain(). Za činnosti v okně je zodpovědná funkce WndProc(), což je realizační funkce tohoto okna. Text uvnitř okna je kreslen při příchodu zprávy WM\_PAINT, fontem odvozeným z tvaru Arial, po získání obrazového kontextu tohoto okna. Ve vhodných situacích je vnitřek okna aktualizován. (Použito volání funkce InvalidateRect(), která vyšle zprávu WM\_PAINT).*



Výběrem z menu jsou aktivována volání dialogových oken, jež mají vlastní realizační funkce.

Dialogová okna jsou vytvářena jako modální, konkrétně voláním funkce `DialogBox()`. Uživatel je tedy veden při zadávání pevným pořadím. To je nutné, neboť mnoho údajů je závislých na svých už zadaných předchůdcích.

Program provádí maximální kontrolu platnosti vkládaných dat. Vkládané hodnoty jsou kontrolovány i ve vazbách na předchozí vložené hodnoty (např. při vkládání jednotlivých objemů při stoupající výšce hladiny program převezme jen vyšší než předchozí objem).

Vzhledem k potencionálně velkému množství případných upozorňujících hlášení na neplatná data, není uživatel na chybný vstup upozorněn samostatným chybovým oknem s textem (voláním funkce `MessageBox()`), ale pouze výstražný zvukovým znamením (voláním funkce `MessageBeep()`).

Chybný vstup je vždy ignorován a dialogové okno otevřeno znovu.

Formální údaje program přebírá ze souboru `formal.txt` v adresáři aplikace. Uživateli je umožněno v případě potřeby tyto údaje změnit, a to volbou Možnosti/Formální údaje :

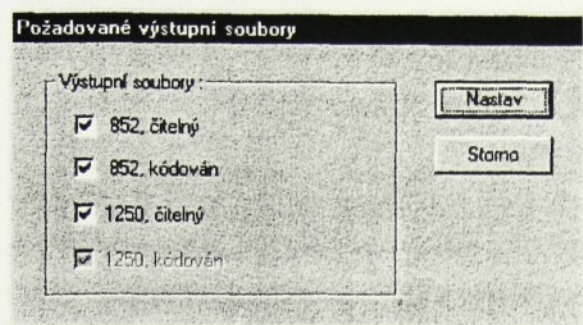
| Formální údaje |                                 |
|----------------|---------------------------------|
| Ulice:         | Rumjancevova 10                 |
| Město:         | 461 30 Liberec, Česká Republika |
| Telefon:       | 048/513 10 72, 510 53 11        |
| Mobil:         | 0602/295 635                    |
| Fax:           | 048/510 42 73                   |
| E-mail:        | graf.skn@geodezie-cs.cz         |
| WWW:           | http://www.geodezie-cs.cz/      |

Obr. 4. Dialog „Formální údaje“

Činnost tohoto dialogového okna má na starosti jeho realizační funkce `FormalDlgProc()`. Ona naplní editační pole známými formálními údaji, použije funkci `SetDlgItemText()`, případně vyzvedne nové údaje pomocí `GetDlgItemText()`.

V případě změny údajů je aktualizován obsah hlavního okna, i souboru `formal.txt`. Uživateli je dále dána možnost ovlivnit počet výstupních souborů programu :





Obr. 5. Dialog „Výstupní soubory“

*Činnost tohoto dialogového okna má na starosti jeho realizační funkce `VystupDlgProc()`. Podle hodnot vyzvednutých v souboru `vystup.txt` inicializuje stav přepínačů (volá funkci `CheckDlgButton()`), případně vyzvedne jejich nový stav (volá funkci `IsDlgButtonChecked()`) a aktualizuje i soubor `vystup.txt`.*

Výstupem programu je zašifrovaný soubor, jehož obsah (po rozšifrování) bude použitelný programem pracujícím s kódovou stránkou 1250.

Tento soubor bude vytvořen vždy, změna jeho výběru je tedy uživateli znemožněna. Navíc je možné ponechat soubor v nezašifrované podobě (1250, čitelný). Dále je možné na přání vytvořit i verze souborů pro prostředí s kódovou stránkou 852, v zašifrované i čitelné formě.

Aktuální konfigurace výstupních souborů je udržována v souboru `vystup.txt`.

Vlastní zadávání údajů se spouští volbou Nádrž/Další. (Opravné zadávání volbou Nádrž/Poslední znovu). Probíhá jako série dialogových oken s jednotlivými dotazy. Vyzvednuté údaje jsou zobrazovány v hlavním okně :



• Kódování údajů pro informační systém

Soubor Nádrž Možnosti O programu...

|                      |                     |
|----------------------|---------------------|
| <b>Nádrž číslo :</b> | <b>1</b>            |
| Název :              | ECONAFTA            |
| Typ nádrže :         | Válcová - stojatá   |
| Kód barvy produktu : | 8000                |
| H max. [cm] :        | 200                 |
| V max. [m3] :        | 20.000000           |
| Mln. plnění, typ :   | výška na konci sání |
| výška [cm] :         | 10                  |
| Max. plnění, typ :   | maximální výška     |
| výška [cm] :         | 190                 |

Obr. 6. Hlavní okno aplikace Koder, stav zadávání údajů

*V průběhu zadávání údajů o nádrži je průběžně vyplňován kontrolní výpis zatím zadaných údajů do vnitřní oblasti hlavního okna. Po úspěšném vyzvednutí informace z některého dialogu je volána funkce `InvalidateRect()`, aby generovala zprávu `WM_PAINT`.*

Při první volbě Nádrž/Další je uživatel nejprve požádán o vložení jména oprávněného uživatele programu :

Zadání oprávněného uživatele programu

Oprávněný uživatel :

[max. 40 znaků]

Ano

Obr. 7. Dialog „Oprávněný uživatel“

*Tento dialog vyřizuje jeho relizační funkce `OpravnenadiDlgProc()`. Text je z editačního pole vyzvednut voláním funkce `GetDlgItemText()`.*

Pokus ukončit dialog bez vložení textu je ignorován.

Při opakované volbě Nádrž/Další se toto okno již nezobrazuje.



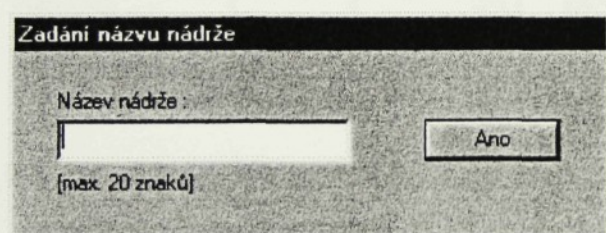
Pro každou nádrž program eviduje :

- název nádrže
- typ nádrže
- kód barvy produktu
- maximální teoretickou výšku  
a kalibrační tabulku nádrže v rozsahu 0 až maximální teoretická výška
- maximální objem
- typ minimálního skutečného plnění
- hodnota minimálního skutečného plnění
- typ maximálního skutečného plnění
- hodnota maximálního skutečného plnění

Vložení každé z informací je vždy úkolem samostatného dialogového okna, požadujícího vždy jedinou informaci. Okna jsou otevírána v pevném pořadí, jsou vytvářena jako nemožná, uživatel je tedy nucen vždy splnit, co okno žádá.

Vyzvednuté údaje jsou v průběhu zadávání doplňovány do tabulky v hlavním okně. Po ukončení celé série dotazů uživatel vidí v hlavním okně všechny zadané údaje o poslední nádrži.

Prvním údajem je název nádrže. Požádá o něj dialogové okno :

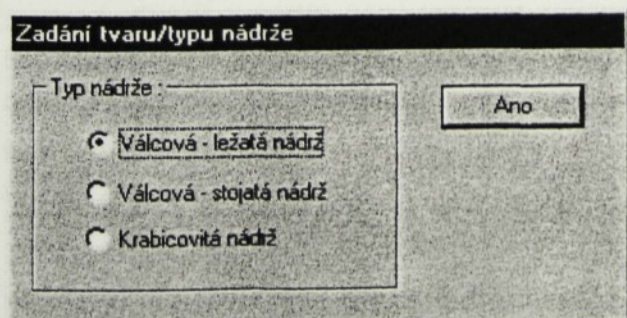


Obr. 8. Dialog „Název nádrže“

*Toto okno má relizační funkci NavezDlgProc().*

Lze zadat max. 20 znaků. Pokus ukončit dialog bez vložení textu bude ignorován.

Dalším údajem je typ nádrže. Požádá o něj dialogové okno :



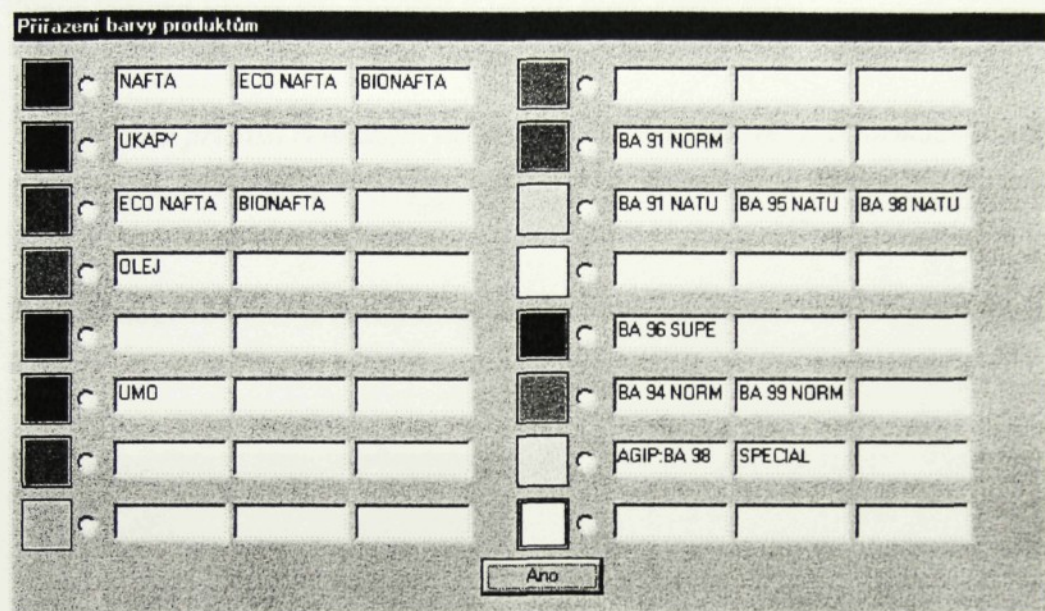
Obr. 9. Dialog „Typ nádrže“



Činnosti v tomto okně má na starost realizační funkce `TypnadobyDlgProc()`. Přepínač je přednastaven na první hodnotě pomocí `CheckRadioButton()`, změny jsou sledovány a při opuštění okna zapamatována poslední poloha.

Při otevření je přednastaven přepínač na první hodnotě.

Dalším údajem je kód barvy, pod kterou bude nádrž přístupná v programu Excaliber. Požádá o něj dialogové okno :



Obr. 10. Dialog „Barva“

Okno zobrazuje 16 skupin prvků.

Každá skupina obsahuje :

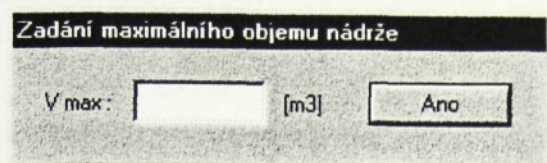
Ikonu barvy, přepínač pro výběr barvy, 3 editační pole pro přiřazení produktů barvě.

Přiřazení barev produktům je programem udržováno v souboru **produkt.txt**, změny budou po zavření dialogu uloženy.

Činnosti v tomto okně vyřizuje realizační funkce `BarvaDlgProc`. Před vykreslením (napojeno na příchod zprávy `WM_INITDIALOG`) je ze souboru `produkt.txt` vyzvednuta aktuální tabulka přiřazení produktů barvám, názvy produktů jsou vyplněny příslušná editační pole (`SetDlgItemText()`). V případě změn jsou tyto vyzvednuty a je aktualizován soubor `produkt.txt`.



Dalším údajem je maximální objem nádrže. Požádá o něj dialogové okno :



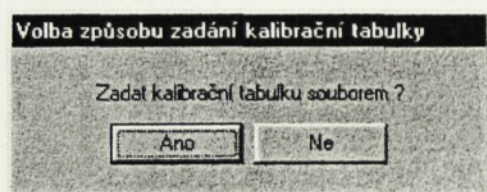
Obr.11. Dialog „Maximální objem“

Pokus ukončit dialog bez vložené hodnoty bude ignorován.

Rovněž tak vložení nevalidních údajů (znaky, záporné číslo) bude ignorováno.

*Oknu přísluší realizační funkce `MaxobjemDlgProc()`. Informace je vyzvednuta z editačního pole jako textový řetězec, ten je konvertován na reálné číslo pomocí funkce `atof()`.*

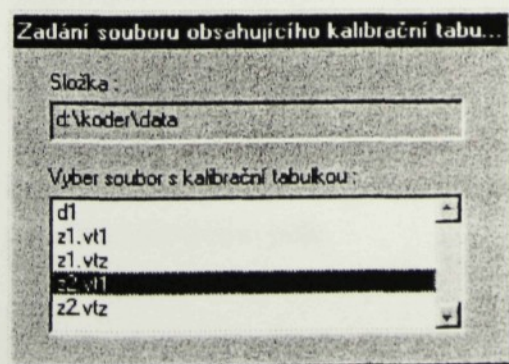
Další dialogové okno je rozhodnutí, zda má být kalibrační tabulka převzata ze souboru, nebo vložena ručně :



Obr. 12. Dialog „Jak zadat“

*Tomuto oknu náleží realizační funkce `JakZadatDlgProc()`.*

Při volbě ANO je vypsán seznam souborů v podadresáři DATA aktuálního adresáře programu :



Obr. 13. Dialog „Soubor“

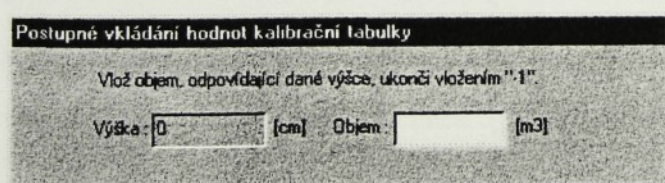


Tomuto oknu přísluší realizační funkce `SouborDlgProc()`. Pro výpis obsahu adresáře je použit prvek seznam (listbox), jež výpisem souborů v podadresáři DATA naplní funkce `DlgDirList()`. Vyzvednutému souboru je prohledáno jeho jméno na výskyt znaků „vz“ signalizujících oficiální formu kalibrační tabulky. Vybraný soubor je otevřen, prohledán za účelem spočítání hodnot kalibrační tabulky a na základě spočtených řádků je alokována paměť pro kalibrační tabulku. Paměť je alokována na globální haldě, pomocí funkce `GlobalAlloc()` a zpřístupněna po uzamčení funkcí `GlobalLock()`.

Program je schopen vyzvednout kalibrační tabulku ze souboru dvěma způsoby.

1. Jde o kalibrační tabulku v oficiální formě dvou souborů **název.vtz** a **název.vt\***, kde **název.vtz** je soubor úvodní, a **název.vt\*** je kalibrační tabulka o třech sloupcích (výška, objem, měrný objem).  
Program rozezná podle znaků „.vt“, že jde o tuto formu dvou souborů a při zvolení kteréhokoli z dvojice se spojí s oběma soubory a vyzvedne vše potřebné.
2. Jde kalibrační tabulku ve formě jen jednoho sloupce s hodnotami objemů, první hodnotu pro výšku 0, který si může uživatel připravit sám.  
Slovo “sloupec” není přesné, data budou viděna ve sloupci, bude-li jako oddělovač použit znak `\n` -odřádkování. Uživatel však může mít čísla oddělená kterýmkoli z obvyklých formátovacích znaků, např. mezerou, tabelátorem.  
Nebude-li v názvu vybraného souboru nalezen řetězec “.vt”, bude program předpokládat, že data jsou pouze hodnotami po sobě jdoucích objemů.

V případě volby ručního zadávání kalibrační tabulky se zobrazí dialogové okno :



Obr. 14. Dialog „Kalibrační tabulka ručně“

Levé editační pole je vyplněno výškou hladiny, pro kterou se očekává vložení objemu do pravého editačního pole.

Toto dialogové okno je vyvoláváno opakovaně, s výškou odstupňovanou po 1 cm.

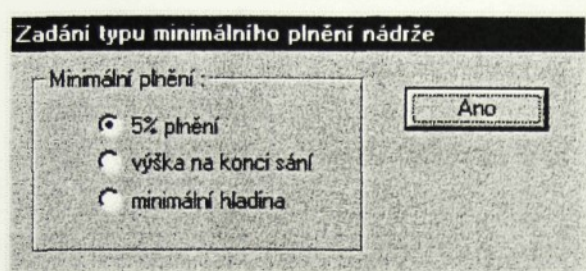
Program provádí kontroly vkládaných údajů.



Nevalidními shledává čísla záporná, čísla menší než jejich předchůdce, čísla větší než zadané  $V_{max}$ . Takové hodnoty jsou ignorovány a program setrvává na stejné hodnotě výšky.

*Tomuto oknu přísluší realizační funkce `KalibrDlgProc()`. Hodnoty jsou vyzvedávány jako řetězce, následně konvertovány na reálná čísla pomocí funkce `atof()`.*

Dalším údajem, evidovaným o nádrži, je počáteční hladina plnění. Nejprve bude uživatel požádán o typ minimálního plnění :

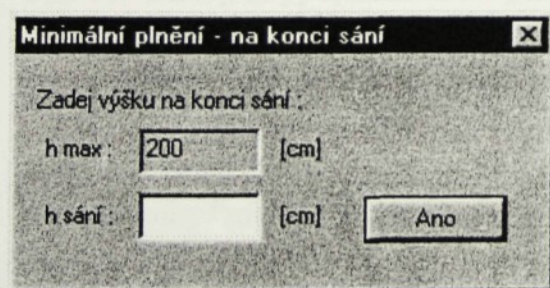


Obr. 15. Dialog „Typ minimálního plnění“

*Tomuto oknu přísluší realizační funkce `MinplnDlgProc()`.*

Při zvolení „5% plnění“ program určí hodnotu výšky sám, v ostatních případech následuje další dotaz.

Po volbě „výška na konci sání“ následuje dotaz na konkrétní hodnotu výšky :

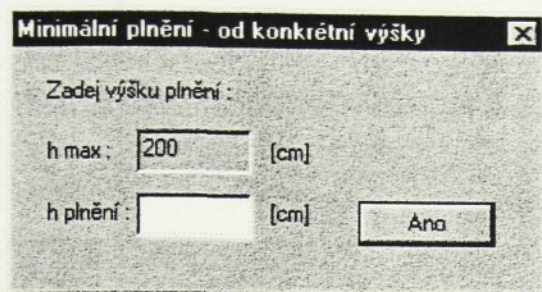


Obr. 16. Dialog „Min. plnění - na konci sání“

*Tomuto oknu náleží realizační funkce `SaniDlgProc()`;*

Po volbě „minimální výška“ též následuje dotaz na konkrétní hodnotu výšky :





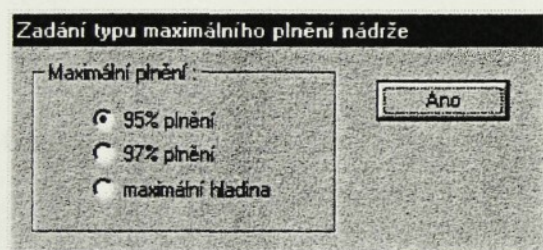
Obr. 17. Dialog „Min. plnění - od konkrétní výšky“

*Tomuto oknu náleží realizační funkce HladMinDlgProc().*

Program provádí obvyklé kontroly.

Navíc dovolí jako počáteční výšku plnění vložit nejvýše polovinu z maximální výšky.

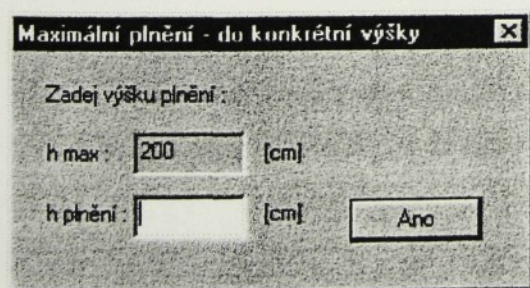
Posledním údajem, evidovaným pro každou nádrž, je horní hladina plnění. Analogicky jako v případě spodní hladiny, nejprve je uživatel tázán na typ horního plnění :



Obr. 18. Dialog „Typ maximálního plnění“

*Tomuto oknu náleží realizační funkce MaxplnDlgProc();*

V případě voleb „95% plnění“ a „97% plnění“ program určí hodnotu horního plnění sám, v případě volby „maximální hladina“ žádá konkrétní výšku :



Obr. 19. Dialog „Max. plnění - do konkrétní výšky“

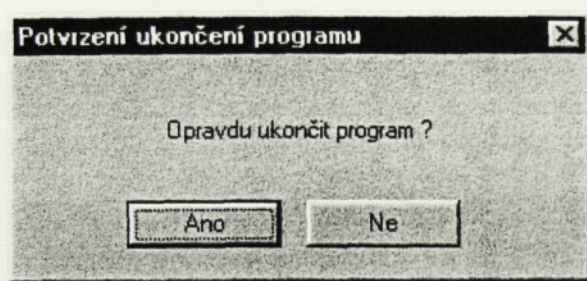


*Tomuto oknu náleží realizační funkce HladMaxDlgProc();*

Nyní uživatel uvidí kontrolní výpis všech zadaných údajů o poslední nádrži, a má na výběr, zda hodnoty poslední nádrže opraví (volba Nádrž/Poslední znovu), přejde k zadávání další nádrže (volba Nádrž/Další), nebo zadávání ukončí a uloží soubory (volba Soubor/Ulož & Zavři).

*Uložení souboru v kódové stránce 1250 zajistí realizační funkce hlavního okna, WndProc(), a to zavoláním kódovacího algoritmu, realizovaného jako funkce KodujProc(). V případě požadavku na soubor v kódové stránce 852 je volána převodní rutina, realizovaná jako funkce PrevedDo852Proc().*

Před úplným ukončením programu je uživatel žádán o potvrzení akce :



Obr. 20. Dialog „Konec“

*Tomuto oknu náleží realizační funkce KonecDlgProc();*

Stiskem "NE" se může uživatel do programu vrátit.

*Přesné řešení jednotlivých situací v programu je zřejmé z výpisu zdrojového textu, viz příloha č. 1 této práce.*



## 5.2. Popis modulu VSTUP

Modul Vstup (konkrétně soubor *vstup.obj*), je určen k přidání do aplikace Excaliber při jejím sestavení. Vzhledem k tomu, že Excaliber je sestaven jako 16-bitová aplikace, bylo nutné modul *vstup.obj* realizovat též jako 16-bitový.

Modul Vstup má zpřístupněny potřebné proměnné Excaliberu pomocí deklarace *extern*.

*Vstup.obj* vznikne přeložením zdrojového textu *vstup.c* do \*.obj kódu. Obsahem zdrojového textu *Vstup.c* je deklarace proměnných a definice funkce *Vstup()*. Program Excaliber provede volání funkce *Vstup()* a tato funkce vykoná své úkoly. Konkrétně provede otevření souboru RESOURCE.RES, jeho dešifrování a naplní vyzvednutými údaji příslušné proměnné.

Přesný popis jak deklarací proměnných je uveden v oddíle 3.1., strana 9. Popis vnitřní organizace souboru je v oddíle 3.2., strana 13. Zdrojový text modulu je v příloze 2. této práce.



## 6. ZÁVĚR

**Mým úkolem v této práci bylo :**

1. Seznámit se s informačním systémem používaným pro kalibrované nádrže.
2. Určit strukturu vstupních dat pro informační systém a způsob jejich zakódování.
3. Sestavit aplikaci na rozhraní Win32 API pro zadávání vstupních dat a generování zakódovaného předávaného souboru. Doplňující požadavek - možnost výběru kódové stránky souboru mezi CP 1250 a CP 852.
4. Ověřit funkčnost aplikace při předání vstupních dat informačnímu systému.

**Řešení úkolů v jednotlivých bodech :**

1. Seznámil jsem se s informačním systémem Excaliber, jeho funkcí a způsobem převzetí vstupních dat. Přesný popis viz oddíl 2., strana 7.
2. Určil jsem strukturu vstupních dat pro Excaliber. Podrobný popis viz oddíl 3.1., strana 9. Navrhl jsem vnitřní uspořádání předávaného souboru. Podrobný popis vnitřního uspořádání viz oddíl 3.2., strana 13. Navrhl jsem algoritmus zakódování souboru. Podrobný popis algoritmu viz oddíl 3.3., strana 14.
3. Seznámil jsem se s mechanismem fungování Windows a aplikačním programovým rozhraním (API). Základní poznatky jsem uvedl v oddíle 3., který začíná na straně 9. Vytvořil jsem aplikaci Koder na rozhraní Win32 API, která generuje vstupní soubory. Aplikace přebere od uživatele údaje o nádržích, uloží je do datových struktur a zapíše do souborů. Převod mezi kódovými stránkami jsem řešil vytvořením převodní tabulky, uložené v souboru, a načítané programem. Popis převodu je v oddíle 3.4., strana 15. K programování jsem použil vývojové prostředí Microsoft C++ 5.0. Podrobný popis funkce programu viz oddíl 5.1., strana 30, výpis zdrojového textu viz příloha č.1.
4. Funkčnost aplikace byla ověřena následovně :  
Vytvořil jsem samostatný modul Vstup(), určený k přidání do aplikace Excaliber. Jeho úkolem je po zavolání jádrem aplikace Excaliber zpětně dekodovat vstupní soubor a



vyzvednutými daty naplnit datové proměnné. Modul vstup jsem přeložil do 16-bitového vstup.obj kódu kvůli kompatibilitě s 16-bitovým programem Excaliber. Autor produktu Excaliber, ing. Michal Moučka, provedl sestavení své aplikace s připojeným modulem Vstup.obj. Popis funkce Vstup je v oddíle 5.2., strana 41. Výpis zdrojového textu je v příloze č. 2.

Předání dat informačnímu systému proběhlo úspěšně, což ověřuje jak správnou funkci programu Koder, tak i správnou funkci modulu Vstup.



## Seznam použité literatury :

- [1] MOUČKA, M. : Sledovací programový systém kalibrovaných nádrží pro čerpací stanice. DP SF - 395, Liberec, TU 1998
- [2] SIMON, R.J. - GOURKER, M - BARNES, B.C. : Win32 API - průvodce vývojáře. Brno, UNIS PUBLISHING 1997.
- [3] RICHTER, J : Windows pro pokročilé a experty. Praha, Computer Press 1997.
- [4] FOŘT, I : Windows 3, techniky programování. Praha, Grada 1992.



```

/*
*****

KODER, verze 2.26

Aplikace pro zadávání vstupních dat pro
informační systém kalibrovaných nádrží
a jejich kódování.

*****
*/
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <winuser.h>

#include "resource.h"

HINSTANCE hInst;
LPCTSTR lpszAppName = "MyApp";
LPCTSTR lpszTitle = "Kódování údajů pro informační systém";
HANDLE hFormal;
HWND hDlgModeless = NULL;
HWND hText = NULL;
char szEditUlice[100],szEditMesto[100],szEditTelefon[100],
      szEditMobil[100],szEditFax[100],szEditMail[100],szEditWww[100];
char szPracovni[60],szPracovniMesto[60],szPracovniTelefon[60],
      szPracovniMobil[60],szPracovniFax[60],szPracovniMail[60],szPracovniWww[60];
char cTemp[2], sTempString[30];
char sZahod[100];
char szPath[255],szPathToData[255];
char sOutNazev[30],sOutTyp[30],sOutBarva[30],sOutHmax[30],sOutVmax[30],
      sOutDoleTyp[30],sOutDoleHodnota[30],sOutNahoreTyp[30],sOutNahoreHodnota[30],
      sOutNadrzCislo[30];
unsigned char iZnaku;
unsigned int uiKoduj;
int i=1;
int iPocitadlo;
int ilkonaRadio = 1;
int iKodovaTabulka[256];
int iZahod;
int iTypNadoby;
int iKalibrVyska = 0;
int i852c, i852k, i1250c;
float fObjemPredchVysky;
float prvni,druha,treti;

BOOL bZadanoOpravnena = FALSE;
BOOL bZadanoNazev=FALSE, bZadanoVmax=FALSE, bZadanoSani=FALSE,
      bZadanoMax=FALSE, bZadanoMin=FALSE;
BOOL bRucneKalibr[10];
BOOL blsTab_MoreColumn[10];
BOOL bSkoncit;
BOOL b852kodovan=FALSE;
BOOL b852citelny=FALSE;
BOOL b1250citelny=FALSE;
BOOL bShowFormal=TRUE;
BOOL bZnovuDialog = TRUE;

typedef struct {
    int typ;int hodnota;
} PLNENI;

int pocet=0,typ_nadrze[9],hmax[9];
unsigned char nazev [9][21],uzivatel[41];
float Qmax[9];
float *Q[10],*Q_temp[10];

```



```
HGLOBAL objem[9];
long barva_produkту[9];
PLNENI dole[9],nahore[9];
unsigned char ulice[30],mesto[40],telefon[30],
        mobil[20],fax[20],mail[40],www[30];

typedef struct{
    unsigned int uiR,uiG,uiB;
    unsigned int pocet;
    unsigned int uiProduktu;
    unsigned char sProdukt1[21],sProdukt2[21],sProdukt3[21];
    unsigned int idEdit1,idEdit2,idEdit3;
    BOOL bVybrana;
} structIKONA;
structIKONA IKONA[17];

DWORD dw;
DWORD dwWrite;

FILE *fr,*fw,*fi,*fo;

/*
*****

                                WinMain

*****

*/

int APIENTRY WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPTSTR lpCmdLine, int nCmdShow)
{
    MSG    msg;
    HWND   hWnd;
    WNDCLASS wc;

    GetCurrentDirectory (254,szPath);
    strcpy(szPathToData,szPath);
    strcat(szPathToData,"\\DATA");

    fi=fopen("vystup.txt","r");
    if(fi!=NULL)
    {
        fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i852c);
        fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i852k);
        fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i1250c);
        fclose(fi);
    }
    else
    {
        MessageBox( hWnd, "Chyba při otevírání VYSTUP.TXT",
            "Pozor :", MB_OK | MB_ICONINFORMATION );
        PostQuitMessage(0);
    }
    fr=fopen("formal.txt","r");
    if(fr!=NULL)
    {
        fgets(szPracovni,50,fr);fgets(szPracovniMesto,50,fr);
        fgets(szPracovniTelefon,50,fr);fgets(szPracovniMobil,50,fr);
        fgets(szPracovniFax,50,fr);fgets(szPracovniMail,50,fr);
        fgets(szPracovniWww,50,fr);
        fclose(fr);
    }
    else
    {
        MessageBox( hWnd, "Chyba při otevírání FORMAL.TXT",
```



```

        "Pozor :", MB_OK | MB_ICONINFORMATION );
    PostQuitMessage(0);
}
fr=fopen("produkt.txt","r");
if(fr!=NULL)fclose(fr);
else
{
    MessageBox( hWnd, "Chyba při otevírání PRODUKT.TXT",
        "Pozor :", MB_OK | MB_ICONINFORMATION );
    PostQuitMessage(0);
}
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovni[iZnaku]));
strncpy(szEditUlice,szPracovni,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniMesto[iZnaku]));
strncpy(szEditMesto,szPracovniMesto,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniTelefon[iZnaku]));
strncpy(szEditTelefon,szPracovniTelefon,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniMobil[iZnaku]));
strncpy(szEditMobil,szPracovniMobil,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniFax[iZnaku]));
strncpy(szEditFax,szPracovniFax,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniMail[iZnaku]));
strncpy(szEditMail,szPracovniMail,iZnaku);
iZnaku=0;do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovniWww[iZnaku]));
strncpy(szEditWww,szPracovniWww,iZnaku);

if(i852c==1) b852citelny=TRUE;else b852citelny=FALSE;
if(i852k==1) b852kodovan=TRUE;else b852kodovan=FALSE;
if(i1250c==1) b1250citelny=TRUE;else b1250citelny=FALSE;

wc.style      = CS_HREDRAW | CS_VREDRAW;
wc.lpfnWndProc = (WNDPROC)WndProc;
wc.cbClsExtra  = 0;
wc.cbWndExtra  = 0;
wc.hInstance   = hInstance;
wc.hIcon       = LoadIcon( hInstance,(HICON)icon2);
wc.hCursor     = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
wc.lpszMenuName = lpszAppName;
wc.lpszClassName = lpszAppName;

if ( !RegisterClass( &wc ) )
    return( FALSE );
hInst = hInstance;

hWnd = CreateWindow( lpszAppName,
    lpszTitle,
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, 0,
    CW_USEDEFAULT, 0,
    NULL,
    NULL,
    hInstance,
    NULL
    );

if ( !hWnd )
    return( FALSE );
ShowWindow( hWnd, nCmdShow );
UpdateWindow( hWnd );
while( GetMessage( &msg, NULL, 0, 0 ) )
{
    TranslateMessage( &msg );
    DispatchMessage( &msg );
}

return( msg.wParam );
}

```



```

/*
*****

                WndProc
        (realizacni fce hlavniho okna)

*****

*/

LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )
{
    int i;
    int ZapisNadrz;
    switch( uMsg )
    {
        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDM_NDR_DAL :
                    if(pocet==9)
                    {
                        MessageBeep(MB_ICONEXCLAMATION);
                        MessageBox( hWnd, "Program umožňuje vložit jen 9 nádrží.",
                        "Pozor :", MB_OK | MB_ICONINFORMATION );
                        break;
                    }
                    pocet++;
                    strcpy(sOutNazev,"");strcpy(sOutTyp,"");
                    strcpy(sOutBarva,"");strcpy(sOutHmax,"");
                    strcpy(sOutVmax,"");
                    strcpy(sOutDoleTyp,"");strcpy(sOutDoleHodnota,"");
                    strcpy(sOutNahoreTyp,"");strcpy(sOutNahoreHodnota,"");
                    sprintf(sOutNadrzCislo,"%d",pocet);
                    bZadanoNazev=FALSE; bZadanoVmax=FALSE;
                    bZadanoSani=FALSE; bZadanoMax=FALSE;
                    bZadanoMin=FALSE;
                    while(bZadanoOpravnena==FALSE)
                    {
                        DialogBox(hInst,"DLG_OPRAVNENA",hWnd,
                        (DLGPROC)OpravnenaDlgProc);
                    }
                    bShowFormal=FALSE;
                    InvalidateRect(hWnd,NULL,TRUE);

                    while(bZadanoNazev==FALSE)
                    {
                        DialogBox(hInst,"DLG_NAZEV",hWnd,
                        (DLGPROC)NazevDlgProc);
                    }
                    InvalidateRect(hWnd,NULL,TRUE);
                    DialogBox(hInst,"DLG_TYPNADOBY",hWnd,
                    (DLGPROC)TypnadDlgProc);
                    InvalidateRect(hWnd,NULL,TRUE);
                    DialogBox(hInst,"DLG_BARVA",hWnd,
                    (DLGPROC)BarvaDlgProc);
                    InvalidateRect(hWnd,NULL,TRUE);

                    while (bZadanoVmax==FALSE)
                    {
                        DialogBox(hInst,"DLG_MAXOBJEM",hWnd,
                        (DLGPROC)MaxobjemDlgProc);
                    }
                    InvalidateRect(hWnd,NULL,TRUE);

                    DialogBox(hInst,"DLG_JAKZADAT",hWnd,

```



```

        (DLGPROC)JakzadatDlgProc);
    InvalidateRect(hWnd,NULL,TRUE);

    if (bRucneKalibr[pocet] == TRUE)
    {
        dw = (DWORD)(300*sizeof(float));
        objem[pocet]=GlobalAlloc(GPTR,dw);
        Q[pocet]=(float*)(GlobalLock(objem[pocet]));
        Q_temp[pocet]=Q[pocet];
        while (bZnovuDialog == TRUE )
        {
            DialogBox(hInst,"DLG_KALIBR",hWnd,
                (DLGPROC)KalibrDlgProc);
        }
        hmax[pocet]=iKalibrVyska;
        sprintf(sOutHmax,"%d",hmax[pocet]);
    }

    else
    {
        SetCurrentDirectory(szPathToData);
        DialogBox(hInst,"DLG_SOUBOR",hWnd,
            (DLGPROC)SouborDlgProc);
    }
    bZnovuDialog = TRUE;/***/rucne i dalsi nadrz
    iKalibrVyska = 0;/***/ dalsi nadrz - rucne - od 0 cm
    SetCurrentDirectory (szPath);
    sprintf(sOutHmax,"%d",hmax[pocet]);
    InvalidateRect(hWnd,NULL,TRUE);

    DialogBox(hInst,"DLG_MINPLN",hWnd,
        (DLGPROC)MinplnDlgProc);
    InvalidateRect(hWnd,NULL,TRUE);
    if(dole[pocet].typ==2)
        while(bZadanoSani==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_SANI",hWnd,
                (DLGPROC)HladsaniDlgProc);
        }

    if(dole[pocet].typ==3)
        while(bZadanoMin==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_MIN",hWnd,
                (DLGPROC)HladminDlgProc);
        }

    InvalidateRect(hWnd,NULL,TRUE);

    DialogBox(hInst,"DLG_MAXPLN",hWnd,
        (DLGPROC)MaxplnDlgProc);
    InvalidateRect(hWnd,NULL,TRUE);
    if(nahore[pocet].typ==3)
        while(bZadanoMax==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_MAX",hWnd,
                (DLGPROC)HladmaxDlgProc);
        }

    InvalidateRect(hWnd,NULL,TRUE);
    break;

case IDM_NDR_ZPET :
    if(bZadanoOpravnena==TRUE)
    {
        GlobalUnlock(Q[pocet]);
        GlobalFree(objem[pocet]);
    }
    else
    {

```



```
MessageBeep(MB_ICONEXCLAMATION);
MessageBox(hWnd, "Dosud nebyla zadána žádná nádrž.",
    "Pozor :", MB_OK | MB_ICONINFORMATION );
break;
}

strcpy(sOutNazev, ""); strcpy(sOutTyp, "");
strcpy(sOutBarva, ""); strcpy(sOutHmax, "");
strcpy(sOutVmax, ""); strcpy(sOutDoleTyp, "");
strcpy(sOutDoleHodnota, ""); strcpy(sOutNahoreTyp, "");
strcpy(sOutNahoreHodnota, "");
sprintf(sOutNadrzCislo, "%d", pocet);

bZadanoNazev=FALSE; bZadanoVmax=FALSE;
bZadanoSani=FALSE; bZadanoMax=FALSE;
bZadanoMin=FALSE;

while(bZadanoOpravena==FALSE)
{
    DialogBox(hInst, "DLG_OPRAVNENA", hWnd,
        (DLGPROC)OpravenaDlgProc);
}
bShowFormal=FALSE;
InvalidateRect(hWnd, NULL, TRUE);

while(bZadanoNazev==FALSE)
{
    DialogBox(hInst, "DLG_NAZEVI", hWnd,
        (DLGPROC)NazevDlgProc);
}
InvalidateRect(hWnd, NULL, TRUE);
DialogBox(hInst, "DLG_TYPNADOBY", hWnd,
    (DLGPROC)TypnadDlgProc);
InvalidateRect(hWnd, NULL, TRUE);

DialogBox(hInst, "DLG_BARVA", hWnd,
    (DLGPROC)BarvaDlgProc);
InvalidateRect(hWnd, NULL, TRUE);

while (bZadanoVmax==FALSE)
{
    DialogBox(hInst, "DLG_MAXOBJEM", hWnd,
        (DLGPROC)MaxobjemDlgProc);
}
InvalidateRect(hWnd, NULL, TRUE);

DialogBox(hInst, "DLG_JAKZADAT", hWnd,
    (DLGPROC)JakzadatDlgProc);
InvalidateRect(hWnd, NULL, TRUE);

if (bRucneKalibr[pocet] == TRUE)
{
    dw = (DWORD)(300*sizeof(float));
    objem[pocet]=GlobalAlloc(GPTR, dw);
    Q[pocet]=(float*)(GlobalLock(objem[pocet]));
    Q_temp[pocet]=Q[pocet];
    while (bZnovuDialog == TRUE )
    {
        DialogBox(hInst, "DLG_KALIBR", hWnd,
            (DLGPROC)KalibrDlgProc);
        iKalibrVyska++;
    }
    hmax[pocet]=iKalibrVyska-2;
    sprintf(sOutHmax, "%d", hmax[pocet]);
}
else
{
```



```

        SetCurrentDirectory(szPathToData);
        DialogBox(hInst,"DLG_SOUBOR",hWnd,
            (DLGPROC)SouborDlgProc);
    }
    bZnovuDialog = TRUE;//pro dalsi nadrz, rucne
    iKalibrVyska = 0;//dalsi nadrz, rucne, od 0 cm
    SetCurrentDirectory (szPath);
    sprintf(sOutHmax,"%d",hmax[pocet]);
    InvalidateRect(hWnd,NULL,TRUE);

    DialogBox(hInst,"DLG_MINPLN",hWnd,
        (DLGPROC)MinplnDlgProc);
    InvalidateRect(hWnd,NULL,TRUE);
    if(dole[pocet].typ==2)
        while(bZadanoSani==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_SANI",hWnd,
                (DLGPROC)HladsaniDlgProc);
        }
    if(dole[pocet].typ==3)
        while(bZadanoMin==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_MIN",hWnd,
                (DLGPROC)HladminDlgProc);
        }
    InvalidateRect(hWnd,NULL,TRUE);

    DialogBox(hInst,"DLG_MAXPLN",hWnd,
        (DLGPROC)MaxplnDlgProc);
    InvalidateRect(hWnd,NULL,TRUE);
    if(nahore[pocet].typ==3)
        while(bZadanoMax==FALSE)
        {
            DialogBox(hInst,"DLG_HLAD_MAX",hWnd,
                (DLGPROC)HladmaxDlgProc);
        }
    InvalidateRect(hWnd,NULL,TRUE);
    break;

case IDM_FORMAL :
    DialogBox( hInst, "DLG_FORMAL", hWnd,
        (DLGPROC)FormalDlgProc );
    InvalidateRect(hWnd,NULL,TRUE);
    break;

case IDM_VYSTUP :
    DialogBox( hInst, "DLG_VYSTUP", hWnd,
        (DLGPROC)VystupDlgProc );
    break;

case IDM_SAVE :
    if(bZadanoOpravnena==TRUE)
    {
        fo=fopen("citelny.125","w");
        fputs(uzivatel,fo);fprintf(fo,"\n");
        fprintf(fo,"%d\n",pocet);
        for(ZapisNadrz=1;ZapisNadrz<=pocet;ZapisNadrz++)
        {
            fputs(nazev[ZapisNadrz],fo);fprintf(fo,"\n");
            fprintf(fo,"%d\n",typ_nadrze[ZapisNadrz]);
            fprintf(fo,"%ld\n",barva_produkту[ZapisNadrz]);
            fprintf(fo,"%d\n",hmax[ZapisNadrz]);
            fprintf(fo,"%f\n",Qmax[ZapisNadrz]);

            Q_temp[ZapisNadrz]=Q[ZapisNadrz];
            for(i=0;i<=hmax[ZapisNadrz];i++)
            {

```



```

        fprintf(fo,"%f\\t",*Q_temp[ZapisNadrz]);
        Q_temp[ZapisNadrz]++;
    }
    fprintf(fo,"\\n");
    Q_temp[ZapisNadrz]=Q[ZapisNadrz];

    fprintf(fo,"%d\\n",dole[ZapisNadrz].typ);
    fprintf(fo,"%d\\n",dole[ZapisNadrz].hodnota);
    fprintf(fo,"%d\\n",nahore[ZapisNadrz].typ);
    fprintf(fo,"%d\\n",nahore[ZapisNadrz].hodnota);
}
fprintf(fo,"%s\\n",szEditUlice);
fprintf(fo,"%s\\n",szEditMesto);
fprintf(fo,"%s\\n",szEditTelefon);
fprintf(fo,"%s\\n",szEditMobil);
fprintf(fo,"%s\\n",szEditFax);
fprintf(fo,"%s\\n",szEditMail);
fprintf(fo,"%s\\n",szEditVww);
fclose(fo);

KodujProc("citelny.125","resource.125");
PrevedDo852Proc("citelny.125","citelny.852");
KodujProc("citelny.852","resource.852");

if(b852kodovan==FALSE)DeleteFile("resource.852");
if(b852citelny==FALSE)DeleteFile("citelny.852");
if(b1250citelny==FALSE)DeleteFile("citelny.125");
}
bSkoncit=FALSE;
DialogBox(hInst,"DLG_KONEC",hWnd,
(DLGPROC)KonecDlgProc);
if(bSkoncit == TRUE) PostQuitMessage(0);

break;

case IDM_ABOUT :
    DialogBox( hInst, "AboutBox", hWnd, (DLGPROC)About );
    break;
}
break;

case WM_PAINT
{
    HDC hDC;
    HPEN hPen;
    HFONT hFont,hFont1,hFont3,hFont4;
    PAINTSTRUCT ps;
    if(bShowFormal==TRUE)
    {
        hFont = CreateFont( 20, 12, 0, 0, FW_NORMAL,
        0, 0, 0, DEFAULT_CHARSET,
        OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS,
        DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN,
        "Arial" );
        hFont1 = CreateFont( 40, 24, 0, 0, FW_NORMAL,
        0, 0, 0, DEFAULT_CHARSET,
        OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS,
        DEFAULT_QUALITY,
        DEFAULT_PITCH | FF_ROMAN,
        "Arial" );
        hDC = BeginPaint(hWnd,&ps);
        SelectObject( hDC, hFont1 );
        TextOut( hDC, 30, 40, "GEODÉZIE ČS, a.s.",
        strlen("GEODÉZIE ČS, a.s."));
    }
}

```



```

DeleteObject(hFont1);

SelectObject( hDC, hFont );
TextOut( hDC, 30, 100, "Ulice",strlen("Ulice"));
TextOut( hDC, 120, 100, ":",strlen(":"));
TextOut( hDC, 150, 100, szEditUlice,
        strlen(szEditUlice));
TextOut( hDC, 30, 130, "Město",strlen("Město"));
TextOut( hDC, 120, 130, ":",strlen(":"));
TextOut( hDC, 150, 130, szEditMesto,
        strlen(szEditMesto));
TextOut( hDC, 30, 160, "Telefon",strlen("Telefon"));
TextOut( hDC, 120, 160, ":",strlen(":"));
TextOut( hDC, 150, 160, szEditTelefon,
        strlen(szEditTelefon));
TextOut( hDC, 30, 190, "Mobil",strlen("Mobil"));
TextOut( hDC, 120, 190, ":",strlen(":"));
TextOut( hDC, 150, 190, szEditMobil,
        strlen(szEditMobil));
TextOut( hDC, 30, 220, "Fax",strlen("Fax"));
TextOut( hDC, 120, 220, ":",strlen(":"));
TextOut( hDC, 150, 220, szEditFax,
        strlen(szEditFax));
TextOut( hDC, 30, 250, "Mail",strlen("Mail"));
TextOut( hDC, 120, 250, ":",strlen(":"));
TextOut( hDC, 150, 250, szEditMail,
        strlen(szEditMail));
TextOut( hDC, 30, 280, "Www",strlen("Www"));
TextOut( hDC, 120, 280, ":",strlen(":"));
TextOut( hDC, 150, 280, szEditWww,strlen(szEditWww));
DeleteObject( hFont );
EndPaint(hWnd,&ps);
}
else
{
    hPen=GetStockObject(BLACK_PEN);
    hFont4 = CreateFont( 28, 18, 0, 0, FW_NORMAL,
    0, 0, 0, DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    CLIP_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN,
    "Arial" );
    hFont3 = CreateFont( 20, 12, 0, 0, FW_NORMAL,
    0, 0, 0, DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    CLIP_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN,
    "Arial" );
    hDC = BeginPaint(hWnd,&ps);
    SelectObject(hDC,hFont4);
    TextOut( hDC,50,50,"Nádrž číslo :",
            strlen("Nádrž číslo :"));
    TextOut( hDC,330,50,sOutNadrzCislo,
            strlen(sOutNadrzCislo));
    DeleteObject(hFont4);
    SelectObject(hDC,hFont3);
    TextOut( hDC, 50, 100, "Název :",strlen("Název :"));
    TextOut(hDC,330,100,sOutNazev,strlen(sOutNazev));
    TextOut( hDC, 50, 125, "Typ nádrže :",
            strlen("Typ nádrže :"));
    TextOut(hDC,330,125,sOutTyp,strlen(sOutTyp));
    TextOut( hDC, 50, 150, "Kód barvy produktu :",
            strlen("Kód barvy produktu :"));
    TextOut(hDC,330,150,sOutBarva,strlen(sOutBarva));
    TextOut( hDC, 50, 175, "H max. [cm] :",

```



```

        strlen("H max. [cm] :");
        TextOut(hDC,330,175,sOutHmax,strlen(sOutHmax));
        TextOut( hDC, 50, 200, "V max. [m3] :",
            strlen("V max. [m3] :"));
        TextOut(hDC,330,200,sOutVmax,strlen(sOutVmax));
        TextOut( hDC, 50, 225, "Min. plnění, typ :",
            strlen("Min. plnění, typ :"));
        TextOut(hDC,330,225,sOutDoleTyp,strlen(sOutDoleTyp));
        TextOut( hDC, 100, 250, "výška [cm] :",
            strlen("výška [cm] :"));
        TextOut(hDC,330,250,sOutDoleHodnota,
            strlen(sOutDoleHodnota));
        TextOut( hDC, 50, 275, "Max. plnění, typ :",
            strlen("Max. plnění, typ :"));
        TextOut(hDC,330,275,sOutNahoreTyp,
            strlen(sOutNahoreTyp));
        TextOut( hDC, 100, 300, "výška [cm] :",
            strlen("výška [cm] :"));
        TextOut(hDC,330,300,sOutNahoreHodnota,
            strlen(sOutNahoreHodnota));
        DeleteObject(hFont3);
        EndPaint(hWnd,&ps);
    }
}
break;

case WM_QUIT :
    for(i=1;i<=pocet;i++)
    {
        GlobalUnlock(Q[i]);
        GlobalFree(objem[i]);
    }
    break;

case WM_DESTROY :
    PostQuitMessage(0);
    break;

default :
    return( DefWindowProc( hWnd, uMsg, wParam, lParam ) );
}
return( 0L );
}

/*
*****
Blok jednotlivych realizacnich fci dialogovych oken
*/

LRESULT CALLBACK FormalDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            SetDlgItemText( hDlg, IDC_EDIT1, szEditUlice);
            SetDlgItemText( hDlg, IDC_EDIT2, szEditMesto);
            SetDlgItemText( hDlg, IDC_EDIT3, szEditTelefon);
            SetDlgItemText( hDlg, IDC_EDIT4, szEditMobil);
            SetDlgItemText( hDlg, IDC_EDIT5, szEditFax);
            SetDlgItemText( hDlg, IDC_EDIT6, szEditMail);
            SetDlgItemText( hDlg, IDC_EDIT7, szEditWww);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {

```



```
case IDOK :
{
    GetDlgItemText( hDlg, IDC_EDIT1,
        szEditUlice, sizeof(szEditUlice)-1 );
    GetDlgItemText( hDlg, IDC_EDIT2,
        szEditMesto, sizeof(szEditMesto)-1 );
    GetDlgItemText( hDlg, IDC_EDIT3,
        szEditTelefon, sizeof(szEditTelefon)-1 );
    GetDlgItemText( hDlg, IDC_EDIT4,
        szEditMobil, sizeof(szEditMobil)-1 );
    GetDlgItemText( hDlg, IDC_EDIT5,
        szEditFax, sizeof(szEditFax)-1 );
    GetDlgItemText( hDlg, IDC_EDIT6,
        szEditMail, sizeof(szEditMail)-1 );
    GetDlgItemText( hDlg, IDC_EDIT7,
        szEditVwww, sizeof(szEditVwww)-1 );
    MoveFile ("formal.txt", "formal.old");

    fw=fopen("formal.txt", "w");
    fputs(szEditUlice, fw); fputs("\n", fw);
    fputs(szEditMesto, fw); fputs("\n", fw);
    fputs(szEditTelefon, fw); fputs("\n", fw);
    fputs(szEditMobil, fw); fputs("\n", fw);
    fputs(szEditFax, fw); fputs("\n", fw);
    fputs(szEditMail, fw); fputs("\n", fw);
    fputs(szEditVwww, fw); fputs("\n", fw);
    fputs("\n", fw);
    fclose(fw);
    EndDialog( hDlg, IDOK );
}
break;

case IDCANCEL :
    EndDialog( hDlg, IDCANCEL );
    break;
}
break;

default :
    return( FALSE );
}

return( TRUE );
}

LRESULT CALLBACK TypnadDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            CheckRadioButton(hDlg, IDC_TYPNAD1, IDC_TYPNAD3, IDC_TYPNAD1);
            typ_nadrze[pocet]=0;

            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(iTypNadoby == 0)
                            strcpy(sOutTyp, "Válcová - ležatá");
                        EndDialog( hDlg, IDOK );
                    }
                    break;
            }
    }
}
```



```

        case IDC_TYPNAD1 :
            typ_nadrze[pocet]=0;
            strcpy(sOutTyp,"Válcová - ležatá");
            break;
        case IDC_TYPNAD2 :
            typ_nadrze[pocet]=1;
            strcpy(sOutTyp,"Válcová - stojatá");
            break;
        case IDC_TYPNAD3 :
            typ_nadrze[pocet]=2;
            strcpy(sOutTyp,"Krabicovitá");
            break;
    }
    break;
default :
    return( FALSE );
}

return( TRUE );
}

LRESULT CALLBACK OpravnaDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            return( TRUE );
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg,IDC_EDIT_OPRAV,uzivatel,40)!=0)
                            bZadanoOpravna=TRUE;
                        else MessageBeep(MB_ICONEXCLAMATION);
                        EndDialog( hDlg, IDOK );
                    }
                    break;
            }
            break;
        default :
            return( FALSE );
    }
    return( TRUE );
}

LRESULT CALLBACK VystupDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            fi=fopen("vystup.txt","r");
            fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i852c);
            fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i852k);
            fscanf(fi,"%s",&sZahod);fscanf(fi,"%d",&i1250c);
            fclose(fi);
            if(i852c==1) b852citelny=TRUE;else b852citelny=FALSE;
            if(i852k==1) b852kodovan=TRUE;else b852kodovan=FALSE;
            if(i1250c==1) b1250citelny=TRUE;else b1250citelny=FALSE;

            CheckDlgButton (hDlg,IDC_CHECK_1250K,BST_CHECKED);
            if(b852citelny==TRUE)

```



```

        CheckDlgButton (hDlg, IDC_CHECK_852C, BST_CHECKED);
        if(b852kodovan==TRUE)
            CheckDlgButton (hDlg, IDC_CHECK_852K, BST_CHECKED);
        if(b1250citelny==TRUE)
            CheckDlgButton (hDlg, IDC_CHECK_1250C, BST_CHECKED);
        return (TRUE);
        break;

case WM_COMMAND :
    switch( LOWORD( wParam ) )
    {
        case IDOK :
            {
                if(IsDlgButtonChecked(hDlg, IDC_CHECK_852C)==BST_CHECKED)
                    b852citelny=TRUE; else b852citelny=FALSE;
                if(IsDlgButtonChecked(hDlg, IDC_CHECK_852K)==BST_CHECKED)
                    b852kodovan=TRUE; else b852kodovan=FALSE;
                if(IsDlgButtonChecked(hDlg, IDC_CHECK_1250C)==BST_CHECKED)
                    b1250citelny=TRUE; else b1250citelny=FALSE;
                MoveFile("vystup.txt", "vystup.old");
                fo=fopen("vystup.txt", "w");
                fprintf(fo, "%s\n", "852CITELNY");
                if(b852citelny==TRUE) fprintf(fo, "%d\n", 1);
                else fprintf(fo, "%d\n", 0);
                fprintf(fo, "%s\n", "852KODOVAN");
                if(b852kodovan==TRUE) fprintf(fo, "%d\n", 1);
                else fprintf(fo, "%d\n", 0);
                fprintf(fo, "%s\n", "1250CITELNY");
                if(b1250citelny==TRUE) fprintf(fo, "%d\n", 1);
                else fprintf(fo, "%d\n", 0);
                fclose(fo);
                EndDialog( hDlg, IDOK );
            }
            break;
        case IDCANCEL :
            EndDialog( hDlg, IDCANCEL );
            break;
    }
    break;
default :
    return( FALSE );
}
return( TRUE );
}

LRESULT CALLBACK NazevDlgProc( HWND hDlg, UINT uMsg,
                               WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg, IDC_EDIT_NAZEV, nazev[pocet], 20)!=0)
                            bZadanoNazev=TRUE;
                        else MessageBeep(MB_ICONEXCLAMATION);
                        strcpy(sOutNazev, nazev[pocet]);
                        EndDialog( hDlg, IDOK );
                    }
                    break;
            }
    }
}

```



```

        break;
    default :
        return( FALSE );
    }
    return( TRUE );
}

LRESULT CALLBACK MaxobjemDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg, IDC_EDIT_MAXOBJEM,
                            sTempString, 20) != 0) bZadanoVmax=TRUE;
                        Qmax[pocet]=(float)atof(sTempString);
                        if(Qmax[pocet]<=0)bZadanoVmax=FALSE;
                        if(bZadanoVmax==TRUE)
                            sprintf(sOutVmax, "%f", Qmax[pocet]);
                        else MessageBeep(MB_ICONEXCLAMATION);
                        EndDialog( hDlg, IDOK );
                    }
                    break;
            }
            break;
        default :
            return( FALSE );
    }
    return( TRUE );
}

LRESULT CALLBACK HladmaxDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            SetDlgItemInt(hDlg, IDC_EDIT_MAX1, hmax[pocet], TRUE);
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg, IDC_EDIT_MAX2,
                            sTempString, 20) != 0) bZadanoMax=TRUE;
                        nahore[pocet].hodnota=(int)atof(sTempString);
                        if((nahore[pocet].hodnota<0)||
                            (nahore[pocet].hodnota>hmax[pocet]))
                            bZadanoMax=FALSE;
                        if(bZadanoMax==TRUE)
                            sprintf(sOutNahoreHodnota, "%d", nahore[pocet].hodnota);
                        else MessageBeep(MB_ICONEXCLAMATION);
                        EndDialog( hDlg, IDOK );
                    }
            }
    }
}

```



```
        break;

    }
    break;
default :
    return( FALSE );
}
return( TRUE );
}

LRESULT CALLBACK HladminDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :

            SetDlgItemInt(hDlg, IDC_EDIT_MIN1, hmax[pocet], TRUE);
            return( TRUE );
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg, IDC_EDIT_MIN2,
                                        sTempString, 20)!=0)bZadanoMin=TRUE;
                        dole[pocet].hodnota=(int)atof(sTempString);
                        if((dole[pocet].hodnota<0)||
                            (dole[pocet].hodnota>hmax[pocet]/2))
                            bZadanoMin=FALSE;
                        if(bZadanoMin==TRUE)
                            sprintf(sOutDoleHodnota, "%d", dole[pocet].hodnota);
                        else MessageBeep(MB_ICONEXCLAMATION);
                        EndDialog( hDlg, IDOK );
                    }
                    break;
            }
            break;
        default :
            return( FALSE );
    }
    return( TRUE );
}

LRESULT CALLBACK HladsaniDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            SetDlgItemInt(hDlg, IDC_EDIT_SANI1, hmax[pocet], TRUE);
            return( TRUE );
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(GetDlgItemText(hDlg, IDC_EDIT_SANI2,
                                        sTempString, 20)!=0)bZadanoSani=TRUE;
                        dole[pocet].hodnota=(int)atof(sTempString);
                        if((dole[pocet].hodnota<0)||
                            (dole[pocet].hodnota>hmax[pocet]/2))
                    }
```



```

        bZadanoSani=FALSE;
        if(bZadanoSani==TRUE)
            sprintf(sOutDoleHodnota,"%d",dole[pocet].hodnota);
        else MessageBeep(MB_ICONEXCLAMATION);
        EndDialog( hDlg, IDOK );
    }
    break;
}
break;
default :
    return( FALSE );
}
return( TRUE );
}

```

```

LRESULT CALLBACK JakzadatDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )

```

```

{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        bRucneKalibr[pocet] = FALSE;
                        EndDialog( hDlg, IDOK );
                    }
                    break;
                case IDCANCEL :
                    bRucneKalibr[pocet] = TRUE;
                    EndDialog( hDlg, IDCANCEL );
                    break;
            }
            break;
        default :
            return( FALSE );
    }
    return( TRUE );
}

```

```

LRESULT CALLBACK MinpInDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )

```

```

{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            CheckRadioButton(hDlg,IDC_MINPLN1,IDC_MINPLN3,IDC_MINPLN1);
            dole[pocet].typ=1;
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(dole[pocet].typ==1)
                        {
                            dole[pocet].hodnota = (int)(0.05*hmax[pocet]);
                            strcpy(sOutDoleTyp,"5% výška");
                        }
                    }
            }
    }
}

```



```

        sprintf(sOutDoleHodnota,"%d",dole[pocet].hodnota);
    }
    EndDialog( hDlg, IDOK );
}
break;
case IDC_MINPLN1 :
    dole[pocet].typ=1;
    strcpy(sOutDoleTyp,"5% výška");
    break;
case IDC_MINPLN2 :
    dole[pocet].typ=2;
    strcpy(sOutDoleTyp,"výška na konci sání");
    break;
case IDC_MINPLN3 :
    dole[pocet].typ=3;
    strcpy(sOutDoleTyp,"minimální výška");
    break;
}
break;
default :
    return( FALSE );
}
return( TRUE );
}
LRESULT CALLBACK MaxPlnDlgProc( HWND hDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            CheckRadioButton(hDlg,IDC_MAXPLN1,IDC_MAXPLN3,IDC_MAXPLN1);
            nahore[pocet].typ=1;
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        if(nahore[pocet].typ==1)
                        {
                            nahore[pocet].hodnota = (int)(0.95*hmax[pocet]);
                            strcpy(sOutNahoreTyp,"95% výška");
                            sprintf(sOutNahoreHodnota,"%d",nahore[pocet].hodnota);
                        }
                        if(nahore[pocet].typ==2)
                        {
                            nahore[pocet].hodnota = (int)(0.97*hmax[pocet]);
                            sprintf(sOutNahoreHodnota,"%d",nahore[pocet].hodnota);
                        }
                    }
                    EndDialog( hDlg, IDOK );
                }
                break;
            case IDC_MAXPLN1 :
                nahore[pocet].typ=1;
                strcpy(sOutNahoreTyp,"95% výška");
                break;
            case IDC_MAXPLN2 :
                nahore[pocet].typ=2;
                strcpy(sOutNahoreTyp,"97% výška");
                break;
            case IDC_MAXPLN3 :
                nahore[pocet].typ=3;
                strcpy(sOutNahoreTyp,"maximální výška");
                break;
            }
    }
}

```



```

        break;
    default :
        return( FALSE );
    }
    return( TRUE );
}
LRESULT CALLBACK BarvaDlgProc( HWND hDlg, UINT uMsg,
                               WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            CheckRadioButton(hDlg, IDC_R1, IDC_R16, IDC_R1);
            IKONA[1].idEdit1=IDC_IK11;
                IKONA[1].idEdit2=IDC_IK12; IKONA[1].idEdit3=IDC_IK13;
            IKONA[2].idEdit1=IDC_IK21;
                IKONA[2].idEdit2=IDC_IK22; IKONA[2].idEdit3=IDC_IK23;
            IKONA[3].idEdit1=IDC_IK31;
                IKONA[3].idEdit2=IDC_IK32; IKONA[3].idEdit3=IDC_IK33;
            IKONA[4].idEdit1=IDC_IK41;
                IKONA[4].idEdit2=IDC_IK42; IKONA[4].idEdit3=IDC_IK43;
            IKONA[5].idEdit1=IDC_IK51;
                IKONA[5].idEdit2=IDC_IK52; IKONA[5].idEdit3=IDC_IK53;
            IKONA[6].idEdit1=IDC_IK61;
                IKONA[6].idEdit2=IDC_IK62; IKONA[6].idEdit3=IDC_IK63;
            IKONA[7].idEdit1=IDC_IK71;
                IKONA[7].idEdit2=IDC_IK72; IKONA[7].idEdit3=IDC_IK73;
            IKONA[8].idEdit1=IDC_IK81;
                IKONA[8].idEdit2=IDC_IK82; IKONA[8].idEdit3=IDC_IK83;
            IKONA[9].idEdit1=IDC_IK91;
                IKONA[9].idEdit2=IDC_IK92; IKONA[9].idEdit3=IDC_IK93;
            IKONA[10].idEdit1=IDC_IK101;
                IKONA[10].idEdit2=IDC_IK102; IKONA[10].idEdit3=IDC_IK103;
            IKONA[11].idEdit1=IDC_IK111;
                IKONA[11].idEdit2=IDC_IK112; IKONA[11].idEdit3=IDC_IK113;
            IKONA[12].idEdit1=IDC_IK121;
                IKONA[12].idEdit2=IDC_IK122; IKONA[12].idEdit3=IDC_IK123;
            IKONA[13].idEdit1=IDC_IK131;
                IKONA[13].idEdit2=IDC_IK132; IKONA[13].idEdit3=IDC_IK133;
            IKONA[14].idEdit1=IDC_IK141;
                IKONA[14].idEdit2=IDC_IK142; IKONA[14].idEdit3=IDC_IK143;
            IKONA[15].idEdit1=IDC_IK151;
                IKONA[15].idEdit2=IDC_IK152; IKONA[15].idEdit3=IDC_IK153;
            IKONA[16].idEdit1=IDC_IK161;
                IKONA[16].idEdit2=IDC_IK162; IKONA[16].idEdit3=IDC_IK163;

            fi=fopen ("produkt.txt","r");
            for(i=1;i<17;i++)
            {
                fscanf(fi,"%d%d%d",&IKONA[i].uiR,&IKONA[i].uiG,&IKONA[i].uiB);
                fscanf(fi,"%d",&IKONA[i].pocet);
                if(IKONA[i].pocet > 0)
                {
                    fgets(szPracovni,50,fi);fgets(szPracovni,50,fi);
                    iZnaku=0;
                    do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovni[iZnaku]));
                    strncpy(IKONA[i].sProdukt1,szPracovni,iZnaku);
                    if(IKONA[i].pocet > 1)
                    {
                        fgets(szPracovni,50,fi);
                        iZnaku=0;
                        do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovni[iZnaku]));
                        strncpy(IKONA[i].sProdukt2,szPracovni,iZnaku);
                        if(IKONA[i].pocet > 2)
                        {
                            fgets(szPracovni,50,fi);
                            iZnaku=0;

```



```

                                do{iZnaku=iZnaku+1;}while(!iscntrl(szPracovni[iZnaku]));
                                strncpy(IKONA[i].sProdukt3,szPracovni,iZnaku);
                            }
                        }
                    }

                for(i=1;i<17;i++)
                {
                    SetDlgItemText(hDlg,IKONA[i].idEdit1,IKONA[i].sProdukt1);
                    SetDlgItemText(hDlg,IKONA[i].idEdit2,IKONA[i].sProdukt2);
                    SetDlgItemText(hDlg,IKONA[i].idEdit3,IKONA[i].sProdukt3);
                }

                fclose(fi);
                return (TRUE);
                break;
            }

case WM_COMMAND :
    switch( LOWORD( wParam ) )
    {
        case IDC_R1 : ilkonaRadio = 1;break;
        case IDC_R2 : ilkonaRadio = 2;break;
        case IDC_R3 : ilkonaRadio = 3;break;
        case IDC_R4 : ilkonaRadio = 4;break;
        case IDC_R5 : ilkonaRadio = 5;break;
        case IDC_R6 : ilkonaRadio = 6;break;
        case IDC_R7 : ilkonaRadio = 7;break;
        case IDC_R8 : ilkonaRadio = 8;break;
        case IDC_R9 : ilkonaRadio = 9;break;
        case IDC_R10 : ilkonaRadio = 10;break;
        case IDC_R11 : ilkonaRadio = 11;break;
        case IDC_R12 : ilkonaRadio = 12;break;
        case IDC_R13 : ilkonaRadio = 13;break;
        case IDC_R14 : ilkonaRadio = 14;break;
        case IDC_R15 : ilkonaRadio = 15;break;
        case IDC_R16 : ilkonaRadio = 16;break;

    }

case IDOK :
    {
        for(i=1;i<17;i++)
        {
            GetDlgItemText (hDlg,IKONA[i].idEdit1,IKONA[i].sProdukt1,20);
            GetDlgItemText (hDlg,IKONA[i].idEdit2,IKONA[i].sProdukt2,20);
            GetDlgItemText (hDlg,IKONA[i].idEdit3,IKONA[i].sProdukt3,20);
        }
        MoveFile("produkt.txt","produkt.old");
        fo=fopen("produkt.txt","w");
        for(i=1;i<17;i++)
        {
            fprintf(fo,"%d\t%d\t%d\n",IKONA[i].uiR,
                    IKONA[i].uiG,IKONA[i].uiB);
            fprintf(fo,"%d\n",IKONA[i].pocet);
            if(IKONA[i].pocet > 0)
            {
                fputs(IKONA[i].sProdukt1,fo);fputs("\n",fo);
                if(IKONA[i].pocet > 1)
                {
                    fputs(IKONA[i].sProdukt2,fo);fputs("\n",fo);
                    if(IKONA[i].pocet > 2)
                    {
                        fputs(IKONA[i].sProdukt3,fo);fputs("\n",fo);
                    }
                }
            }
        }
    }

    fclose(fo);

```



```

        barva_produkту[pocet]=RGB(IKONA[iIkonaRadio].uiR,
                                   IKONA[iIkonaRadio].uiG,IKONA[iIkonaRadio].uiB);
        sprintf(sOutBarva,"%X",barva_produkту[pocet]);
        EndDialog( hDlg, IDOK );
    }
    break;
}
break;
default :
    return( FALSE );
}
return( TRUE );
}

```

```

LRESULT CALLBACK SouborDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )

```

```

{
    char szTmp[30]="";
    char szTmp1[30]="";
    char szTmp2[30]="";
    float fZahod,fObjem;
    char sZahod[100];
    int iStatus;
    int i;
    switch( uMsg )
    {
        case WM_INITDIALOG :
            DlgDirList( hDlg, "*.**", IDC_LIST, IDC_DIRECTORY, 0 );
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDC_LIST :
                    if ( HIWORD( wParam ) == LBN_DBLCLK )
                    {
                        DlgDirSelectEx( hDlg, szTmp, 200, IDC_LIST );

                        if(strstr(szTmp, ".vt")!=NULL)
                        {
                            blsTab_MoreColumn[pocet]=TRUE;
                            for (i=0;i<(int)strlen(szTmp)-1;i++)
                            {
                                if(szTmp[i]=='.')
                                {
                                    strncpy(szTmp1,szTmp,i);
                                    strncpy(szTmp2,szTmp,i);
                                    strcat(szTmp1, ".vt1");
                                    strcat(szTmp2, ".vtz");
                                }
                            }
                        }
                    }
                    else
                    {
                        blsTab_MoreColumn[pocet]=FALSE;
                        strcpy(szTmp1,szTmp);
                    }

                    fi = fopen(szTmp1,"r");
                    iPocitadlo = 0;
                    do
                    {
                        iStatus = fscanf(fi,"%f",&fZahod);
                        iPocitadlo++;
                    }
                    while (iStatus!=EOF);

```



```

iPocitadlo--; //posledni skoncil EOF a nenese hodnotu
fclose(fi);

if(bIsTab_MoreColumn[pocet]==TRUE)
{
    hmax[pocet]=(int)((iPocitadlo+1)/3);
    dw = (DWORD)((hmax[pocet]+1)*sizeof(float));
    objem[pocet]=GlobalAlloc(GPTR,dw);
    Q[pocet]=(float*)(GlobalLock(objem[pocet]));
    Q_temp[pocet]=Q[pocet];
    fi=fopen(szTmp2,"r");
    fgets(sZahod,99,fi);fgets(sZahod,99,fi);
    fgets(sZahod,99,fi);

    fscanf(fi,"%s",&sZahod);fscanf(fi,"%s",&sZahod);
    fscanf(fi,"%f",&fObjem); // objem pro vysku 0

    fscanf(fi,"%s",&sZahod);fscanf(fi,"%s",&sZahod);
    fscanf(fi,"%f",&Qmax[pocet]); //teoreticky max. V
    fclose(fi);
    Q_temp[pocet]=Q[pocet];
    *Q_temp[pocet]=fObjem;
    Q_temp[pocet]++;

    // potom v_1 az v_hmax ze souboru *.vt3 - 3sloup.
    //*****
    fi=fopen(szTmp1,"r");
    for (i=1;i<=hmax[pocet];i++)
    {
        fscanf(fi,"%f",&fZahod);
        fscanf(fi,"%f",&fObjem);
        *Q_temp[pocet]=fObjem;
        Q_temp[pocet]++;
        fscanf(fi,"%f",&fZahod);
    }
    Q_temp[pocet]=Q[pocet];
    fclose(fi);
}
else
{
    //cti jeden sloupec z *. ukladej ho od v_0
    //*****

    hmax[pocet]=(iPocitadlo-1);
    //1_sloupec: Poc.od 0,h souhlasi
    dw = (DWORD)((hmax[pocet]+1)*sizeof(float));
    objem[pocet]=GlobalAlloc(GPTR,dw);

    Q[pocet]=(float*)(GlobalLock(objem[pocet]));
    Q_temp[pocet]=Q[pocet];
    fi=fopen(szTmp1,"r");
    for(i=0;i<=hmax[pocet];i++)
    {
        fscanf(fi,"%f",&prvni);
        *Q_temp[pocet]=prvni;
        *Q_temp[pocet]++;
    }
    Q_temp[pocet]=Q[pocet];
    fclose(fi);
}

Q_temp[pocet]=Q[pocet];
EndDialog( hDlg, TRUE );
}

break;

```



```
    }
    break;

default :
    return( FALSE );
}

return( TRUE );
}

LRESULT CALLBACK KalibrDlgProc( HWND hDlg, UINT uMsg,
                                WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            SetDlgItemInt (hDlg, IDC_KALIBR_VYSKA, iKalibrVyska, TRUE);
            return (TRUE);
            break;

        case WM_COMMAND:
            if ( LOWORD(wParam) == IDOK )
            {
                if(iKalibrVyska>0)
                {
                    Q_temp[pocet]-;fObjemPredchVysky=*Q_temp[pocet];
                    Q_temp[pocet]++;
                }
                GetDlgItemText(hDlg, IDC_KALIBR_OBJEM, sTempString, 29);
                *Q_temp[pocet]=(float)(atof(sTempString));
                if(*Q_temp[pocet]==(double)(-1))bZnovuDialog=FALSE;

                if(bZnovuDialog)
                {
                    if(iKalibrVyska>1)
                        if(*Q_temp[pocet]<fObjemPredchVysky)
                            {iKalibrVyska--;Q_temp[pocet]-;}

                    if(bZnovuDialog)if(*Q_temp[pocet]>Qmax[pocet])
                        {iKalibrVyska--;Q_temp[pocet]-;}

                    if(bZnovuDialog==TRUE)
                    {
                        iKalibrVyska++;
                        Q_temp[pocet]++;
                    }
                    else
                    {
                        MessageBeep(MB_ICONEXCLAMATION);
                        iKalibrVyska--;
                    }
                }

                EndDialog(hDlg, TRUE);
                return (TRUE);
            }
            break;

        default :
            return( FALSE );
    }
    return( TRUE );
}

LRESULT CALLBACK About( HWND hDlg,
                        UINT message,
                        WPARAM wParam,
                        LPARAM lParam)
{

```



```

switch (message)
{
    case WM_INITDIALOG:
        return (TRUE);

    case WM_COMMAND:
        if ( LOWORD(wParam) == IDOK
            || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, TRUE);
            return (TRUE);
        }
        break;
}
return (FALSE);
}

LRESULT CALLBACK KonecDlgProc( HWND hDlg, UINT uMsg,
                              WPARAM wParam, LPARAM lParam )
{
    switch( uMsg )
    {
        case WM_INITDIALOG :
            return (TRUE);
            break;

        case WM_COMMAND :
            switch( LOWORD( wParam ) )
            {
                case IDOK :
                    {
                        bSkoncit = TRUE;
                        EndDialog( hDlg, IDOK );
                    }
                    break;
                case IDCANCEL :
                    {
                        bSkoncit = FALSE;
                        EndDialog( hDlg, IDCANCEL );
                    }
                    break;
            }
            break;
        default :
            return( FALSE );
    }
    return( TRUE );
}

/* Konec bloku realizacnich fci dialogovych oken
*****
*/

/*
*****
Sifrovaci funkce
*****
*/

void KodujProc(const char *fZdrojovy,const char *fCilovy)
{
    fi=fopen(fZdrojovy,"rb");
    fo=fopen(fCilovy,"wb");
    while((uiKoduj = fgetc(fi)) != EOF)

```



```

    {
        if (uiKoduj > 127)
        {
            if(uiKoduj==255)
            {
                fputc(03,fo);
            }
            else
            {
                fputc(02,fo);
                fputc(uiKoduj+1,fo);
            }
        }
        else
            fputc(uiKoduj*2,fo);
    }
    fclose(fi);
    fclose(fo);
}

/*
*****
Prevodni funkce CP 1250 --> CP 852
*****
*/

void PrevedDo852Proc(const char *fZdrojovy,const char *fCilovy)
{
    for (i=0;i<128;i++) iKodovaTabulka[i] = i;
    fi = fopen("1250852.txt","r");
    for (i=128;i<256;i++) fscanf(fi,"%d%d",&iZahod,&iKodovaTabulka[i]);
    fclose(fi);
    fi = fopen(fZdrojovy,"rb");
    fo = fopen(fCilovy,"wb");
    while((uiKoduj = fgetc(fi)) != EOF) fputc(iKodovaTabulka[uiKoduj],fo);
    fclose(fi);
    fclose(fo);
}

```



/\*

\*\*\*\*\*

VSTUP  
Verze 2.02

Modul pro dekodování souboru a vyzvednutí  
dat ze souboru.

\*\*\*\*\*

\*/

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <alloc.h>
#include <windows.h>
#include <ctype.h>
```

```
typedef struct {
    int typ; int hodnota;
} PLNENI ;
```

```
extern int pocet, typ_nadrze[9], hmax[9] ;
extern unsigned char nazev[9][21], uzivatel[41] ;
extern float Qmax[9];
extern float far *Q[9];
```

```
extern HGLOBAL objem[9] ;
extern long barva_produkту[9] ;
extern PLNENI dole[9], nahore[9] ;
extern unsigned char ulice[30], mesto[40], telefon[30], mobil[20],
    fax[20], mail[40], www[30] ;
```

```
float far *Q_temp[9];
```

```
    unsigned char iZnaku;
    int i,ZapisNadrz;
    unsigned char sPracovni[60];
    FILE *fi,*fo;
    DWORD dw ;
    unsigned int iKoduj;
    float fObjem;
```

```
int Vstup ( void )
```

```
{
    fi = fopen ( "resource.res", "rb" ) ;
    fo = fopen ( "temp.txt","wb" );
    if ( fi == NULL ) return ( 1 ) ;

    while((iKoduj = fgetc(fi)) != EOF)
    {
        if(iKoduj == 02)
        {
            iKoduj = fgetc(fi);
            fputc(iKoduj-1,fo);
        }
        else
        {
            if(iKoduj == 03)
            {
                fputc(255,fo);
            }
            else
            {
                fputc(iKoduj/2,fo);
            }
        }
    }
}
```



```

    }
}

fclose(fi);
fclose(fo);

fi = fopen("temp.txt","r");

fgets(sPracovni,49,fi);
iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>40) iZnaku=40;
strncpy(uzivatel,sPracovni,iZnaku);/*****opraveny uzivatel

fscanf(fi,"%d",&pocet);/*****pocet nadrzi

for(ZapisNadrz=0;ZapisNadrz<pocet;ZapisNadrz++)
{
//ZapisNadrz=0;
fgets(sPracovni,50,fi);fgets(sPracovni,50,fi);
iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>40) iZnaku=40;
strncpy(nazev[ZapisNadrz],sPracovni,iZnaku);/*****nazev nadrze

fscanf(fi,"%d",&typ_nadrze[ZapisNadrz]);/*****typ nadrze

fscanf(fi,"%ld",&barva_produkту[ZapisNadrz]);/*****kod barvy
fscanf(fi,"%d",&hmax[ZapisNadrz]);/*****hmax
fscanf(fi,"%f",&Qmax[ZapisNadrz]);/*****qmax

dw = (DWORD)((hmax[ZapisNadrz]+1) * sizeof (float) );
objem[ZapisNadrz] = GlobalAlloc ( GPTR , dw );
if ( objem[ZapisNadrz] == NULL ) return (2) ;
Q[ZapisNadrz] = (float *) GlobalLock ( objem[ZapisNadrz] );
Q_temp[ZapisNadrz]=Q[ZapisNadrz];

for(i=0;i<=hmax[ZapisNadrz];i++)
{
    fscanf(fi,"%f",&fObjem);

    *Q_temp[ZapisNadrz]=fObjem;
    Q_temp[ZapisNadrz]++;
}

    Q_temp[ZapisNadrz]=Q[ZapisNadrz];
    fscanf(fi,"%d",&dole[ZapisNadrz].typ);/*****dole.typ

fscanf(fi,"%d",&dole[ZapisNadrz].hodnota);/*****dole.hodnota

fscanf(fi,"%d",&nahore[ZapisNadrz].typ);/*****nahore.typ

fscanf(fi,"%d",&nahore[ZapisNadrz].hodnota);/*****nahore.hodnota
}
fgets(sPracovni,49,fi);
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>29) iZnaku=29;
strncpy(ulice,sPracovni,iZnaku);/*****ulice

fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>39) iZnaku=39;
strncpy(mesto,sPracovni,iZnaku);/*****mesto
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>29) iZnaku=29;
strncpy(telefon,sPracovni,iZnaku);/*****telefon

```



```
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>19) iZnaku=19;
strncpy(mobil,sPracovni,iZnaku);/*****mobil
```

```
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>19) iZnaku=19;
strncpy(fax,sPracovni,iZnaku);/*****fax
```

```
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>39) iZnaku=39;
strncpy(mail,sPracovni,iZnaku);/*****mail
```

```
fgets(sPracovni,49,fi);iZnaku=0;
do{iZnaku=iZnaku+1;}while(!iscntrl(sPracovni[iZnaku]));
if(iZnaku>29) iZnaku=29;
strncpy(www,sPracovni,iZnaku);/*****www
fclose(fi);
```

```
DeleteFile("temp.txt");
```

```
return 0 ;
}
```



Tabulka 2a. ASCII kódy 0 - 127 :

| Dec | Hex | Char | Ctrl +   | ASCII               | Dec | Hex | Char | Ctrl +   | ASCII                     |
|-----|-----|------|----------|---------------------|-----|-----|------|----------|---------------------------|
| 0   | 00  | NUL  | CTRL + @ | null                | 16  | 10  | DLE  | CTRL + P | data link escape          |
| 1   | 01  | SOH  | CTRL + A | start of heading    | 17  | 11  | DC1  | CTRL + Q | device control 1 X-ON     |
| 2   | 02  | STX  | CTRL + B | start of text       | 18  | 12  | DC2  | CTRL + R | device control 2 TAPE     |
| 3   | 03  | ETX  | CTRL + C | end of text         | 19  | 13  | DC3  | CTRL + S | device control 3 X-OFF    |
| 4   | 04  | EOT  | CTRL + D | end of transmission | 20  | 14  | DC4  | CTRL + T | device control 4 TAPE     |
| 5   | 05  | ENQ  | CTRL + E | enquiry             | 21  | 15  | NAK  | CTRL + U | negative acknowledge      |
| 6   | 06  | ACK  | CTRL + F | acknowledge         | 22  | 16  | SYN  | CTRL + V | synchronous idle          |
| 7   | 07  | BEL  | CTRL + G | bell                | 23  | 17  | ETB  | CTRL + W | end of transmission block |
| 8   | 08  | BS   | CTRL + H | backspace           | 24  | 18  | CAN  | CTRL + X | cancel line               |
| 9   | 09  | HT   | CTRL + I | tab horizontally    | 25  | 19  | EM   | CTRL + Y | end of medium             |
| 10  | 0A  | LF   | CTRL + J | line feed           | 26  | 1A  | SUB  | CTRL + Z | substitute                |
| 11  | 0B  | VT   | CTRL + K | tab vertically      | 27  | 1B  | ESC  | CTRL + [ | escape                    |
| 12  | 0C  | FF   | CTRL + L | form feed           | 28  | 1C  | FS   | CTRL + \ | file separator            |
| 13  | 0D  | CR   | CTRL + M | carriage return     | 29  | 1D  | GS   | CTRL + ] | group separator           |
| 14  | 0E  | SO   | CTRL + N | shift out           | 30  | 1E  | RS   | CTRL + ^ | record separator          |
| 15  | 0F  | SI   | CTRL + O | shift in            | 31  | 1F  | US   | CTRL + _ | unit separator            |

| Dec | Hex | Char     | Dec | Hex | Char | Dec | Hex | Char |
|-----|-----|----------|-----|-----|------|-----|-----|------|
| 32  | 20  | (mezera) | 64  | 40  | @    | 96  | 60  | `    |
| 33  | 21  | !        | 65  | 41  | A    | 97  | 61  | a    |
| 34  | 22  | "        | 66  | 42  | B    | 98  | 62  | b    |
| 35  | 23  | #        | 67  | 43  | C    | 99  | 63  | c    |
| 36  | 24  | \$       | 68  | 44  | D    | 100 | 64  | d    |
| 37  | 25  | %        | 69  | 45  | E    | 101 | 65  | e    |
| 38  | 26  | &        | 70  | 46  | F    | 102 | 66  | f    |
| 39  | 27  | '        | 71  | 47  | G    | 103 | 67  | g    |
| 40  | 28  | (        | 72  | 48  | H    | 104 | 68  | h    |
| 41  | 29  | )        | 73  | 49  | I    | 105 | 69  | i    |
| 42  | 2A  | *        | 74  | 4A  | J    | 106 | 6A  | j    |
| 43  | 2B  | +        | 75  | 4B  | K    | 107 | 6B  | k    |
| 44  | 2C  | ,        | 76  | 4C  | L    | 108 | 6C  | l    |
| 45  | 2D  | -        | 77  | 4D  | M    | 109 | 6D  | m    |
| 46  | 2E  | .        | 78  | 4E  | N    | 110 | 6E  | n    |
| 47  | 2F  | /        | 79  | 4F  | O    | 111 | 6F  | o    |
| 48  | 30  | 0        | 80  | 50  | P    | 112 | 70  | p    |
| 49  | 31  | 1        | 81  | 51  | Q    | 113 | 71  | q    |
| 50  | 32  | 2        | 82  | 52  | R    | 114 | 72  | r    |
| 51  | 33  | 3        | 83  | 53  | S    | 115 | 73  | s    |
| 52  | 34  | 4        | 84  | 54  | T    | 116 | 74  | t    |
| 53  | 35  | 5        | 85  | 55  | U    | 117 | 75  | u    |
| 54  | 36  | 6        | 86  | 56  | V    | 118 | 76  | v    |
| 55  | 37  | 7        | 87  | 57  | W    | 119 | 77  | w    |
| 56  | 38  | 8        | 88  | 58  | X    | 120 | 78  | x    |
| 57  | 39  | 9        | 89  | 59  | Y    | 121 | 79  | y    |
| 58  | 3A  | :        | 90  | 5A  | Z    | 122 | 7A  | z    |
| 59  | 3B  | ;        | 91  | 5B  | [    | 123 | 7B  | {    |
| 60  | 3C  | <        | 92  | 5C  | \    | 124 | 7C  |      |
| 61  | 3D  | =        | 93  | 5D  | ]    | 125 | 7D  | }    |
| 62  | 3E  | >        | 94  | 5E  | ^    | 126 | 7E  | ~    |
| 63  | 3F  | ?        | 95  | 5F  | _    | 127 | 7F  |      |



Tabulka 2b. ASCII kódy 127 -255 :

| Dec | Hex | Printable | Printable | Printable | Printable |
|-----|-----|-----------|-----------|-----------|-----------|
| 128 | 80  | Ç         | Ç         | Ç         |           |
| 129 | 81  | à         | à         | à         |           |
| 130 | 82  | á         | á         | á         |           |
| 131 | 83  | ä         | ä         | ä         |           |
| 132 | 84  | å         | å         | å         |           |
| 133 | 85  | Ä         | Ä         | Ä         |           |
| 134 | 86  | Å         | Å         | Å         |           |
| 135 | 87  | ç         | ç         | ç         |           |
| 136 | 88  | è         | è         | è         |           |
| 137 | 89  | é         | É         | é         | %         |
| 138 | 8A  | ê         | Ê         | ê         | \$        |
| 139 | 8B  | ë         | Ë         | ë         | ¢         |
| 140 | 8C  | ì         | Ì         | ì         | £         |
| 141 | 8D  | í         | Í         | í         | ¢         |
| 142 | 8E  | Î         | Î         | Î         | £         |
| 143 | 8F  | Ï         | Ï         | Ï         | £         |
| 144 | 90  | Ê         | Ê         | Ê         |           |
| 145 | 91  | æ         | æ         | æ         |           |
| 146 | 92  | Æ         | Æ         | Æ         |           |
| 147 | 93  | ó         | ó         | ó         |           |
| 148 | 94  | ô         | ô         | ô         |           |
| 149 | 95  | õ         | Õ         | õ         |           |
| 150 | 96  | ü         | Ü         | ü         |           |
| 151 | 97  | û         | Û         | û         |           |
| 152 | 98  | ÿ         | ÿ         | ÿ         |           |
| 153 | 99  | Ö         | Ö         | Ö         | ™         |
| 154 | 9A  | Ù         | Ù         | Ù         | £         |
| 155 | 9B  | Ú         | Ú         | Ú         | £         |
| 156 | 9C  | Û         | Û         | Û         | £         |
| 157 | 9D  | Ü         | Ü         | Ü         | £         |
| 158 | 9E  | Ý         | Ý         | Ý         | £         |
| 159 | 9F  | ÿ         | ÿ         | ÿ         | £         |
| 160 | A0  | À         | À         | À         |           |
| 161 | A1  | Á         | Á         | Á         |           |
| 162 | A2  | Â         | Â         | Â         |           |
| 163 | A3  | Ã         | Ã         | Ã         | L         |
| 164 | A4  | Ä         | Ä         | Ä         | £         |
| 165 | A5  | Å         | Å         | Å         | A         |
| 166 | A6  | Æ         | Æ         | Æ         | Z         |
| 167 | A7  | Ç         | Ç         | Ç         | £         |
| 168 | A8  | È         | È         | È         | £         |
| 169 | A9  | É         | É         | É         | £         |
| 170 | AA  | Ê         | Ê         | Ê         | £         |
| 171 | AB  | Ë         | Ë         | Ë         | £         |
| 172 | AC  | Ì         | Ì         | Ì         | £         |
| 173 | AD  | Í         | Í         | Í         | £         |
| 174 | AE  | Î         | Î         | Î         | £         |
| 175 | AF  | Ï         | Ï         | Ï         | £         |
| 176 | B0  | Ñ         | Ñ         | Ñ         | £         |
| 177 | B1  | Ò         | Ò         | Ò         | £         |
| 178 | B2  | Ó         | Ó         | Ó         | £         |
| 179 | B3  | Ô         | Ô         | Ô         | £         |
| 180 | B4  | Õ         | Õ         | Õ         | £         |
| 181 | B5  | Ö         | Ö         | Ö         | £         |
| 182 | B6  | ×         | ×         | ×         | £         |
| 183 | B7  | Ø         | Ø         | Ø         | £         |
| 184 | B8  | Ù         | Ù         | Ù         | £         |
| 185 | B9  | Ú         | Ú         | Ú         | £         |
| 186 | BA  | Û         | Û         | Û         | £         |
| 187 | BB  | Ü         | Ü         | Ü         | £         |
| 188 | BC  | Ý         | Ý         | Ý         | £         |
| 189 | BD  | Þ         | Þ         | Þ         | £         |
| 190 | BE  | ß         | ß         | ß         | £         |
| 191 | BF  | à         | à         | à         | £         |
| 192 | C0  | Á         | Á         | Á         | £         |
| 193 | C1  | Â         | Â         | Â         | £         |
| 194 | C2  | Ã         | Ã         | Ã         | £         |
| 195 | C3  | Ä         | Ä         | Ä         | £         |
| 196 | C4  | Å         | Å         | Å         | £         |
| 197 | C5  | Æ         | Æ         | Æ         | £         |
| 198 | C6  | Ç         | Ç         | Ç         | £         |
| 199 | C7  | È         | È         | È         | £         |
| 200 | C8  | É         | É         | É         | £         |
| 201 | C9  | Ê         | Ê         | Ê         | £         |
| 202 | CA  | Ë         | Ë         | Ë         | £         |
| 203 | CB  | Ì         | Ì         | Ì         | £         |
| 204 | CC  | Í         | Í         | Í         | £         |
| 205 | CD  | Î         | Î         | Î         | £         |
| 206 | CE  | Ï         | Ï         | Ï         | £         |
| 207 | CF  | Ñ         | Ñ         | Ñ         | £         |
| 208 | D0  | Ò         | Ò         | Ò         | £         |
| 209 | D1  | Ó         | Ó         | Ó         | £         |
| 210 | D2  | Ô         | Ô         | Ô         | £         |
| 211 | D3  | Õ         | Õ         | Õ         | £         |
| 212 | D4  | Ö         | Ö         | Ö         | £         |
| 213 | D5  | ×         | ×         | ×         | £         |
| 214 | D6  | Ø         | Ø         | Ø         | £         |
| 215 | D7  | Ù         | Ù         | Ù         | £         |
| 216 | D8  | Ú         | Ú         | Ú         | £         |
| 217 | D9  | Û         | Û         | Û         | £         |
| 218 | DA  | Ü         | Ü         | Ü         | £         |
| 219 | DB  | Ý         | Ý         | Ý         | £         |
| 220 | DC  | Þ         | Þ         | Þ         | £         |
| 221 | DD  | ß         | ß         | ß         | £         |
| 222 | DE  | à         | à         | à         | £         |
| 223 | DF  | á         | á         | á         | £         |
| 224 | E0  | â         | â         | â         | £         |
| 225 | E1  | ã         | ã         | ã         | £         |
| 226 | E2  | ä         | ä         | ä         | £         |
| 227 | E3  | å         | å         | å         | £         |
| 228 | E4  | æ         | æ         | æ         | £         |
| 229 | E5  | ç         | ç         | ç         | £         |
| 230 | E6  | è         | è         | è         | £         |
| 231 | E7  | é         | é         | é         | £         |
| 232 | E8  | ê         | ê         | ê         | £         |
| 233 | E9  | ë         | ë         | ë         | £         |
| 234 | EA  | ì         | ì         | ì         | £         |
| 235 | EB  | í         | í         | í         | £         |
| 236 | EC  | î         | î         | î         | £         |
| 237 | ED  | ï         | ï         | ï         | £         |
| 238 | EE  | ñ         | ñ         | ñ         | £         |
| 239 | EF  | ò         | ò         | ò         | £         |
| 240 | F0  | ó         | ó         | ó         | £         |
| 241 | F1  | ô         | ô         | ô         | £         |
| 242 | F2  | õ         | õ         | õ         | £         |
| 243 | F3  | ö         | ö         | ö         | £         |
| 244 | F4  | ÷         | ÷         | ÷         | £         |
| 245 | F5  | ø         | ø         | ø         | £         |
| 246 | F6  | ù         | ù         | ù         | £         |
| 247 | F7  | ú         | ú         | ú         | £         |
| 248 | F8  | û         | û         | û         | £         |
| 249 | F9  | ü         | ü         | ü         | £         |
| 250 | FA  | ý         | ý         | ý         | £         |
| 251 | FB  | þ         | þ         | þ         | £         |
| 252 | FC  | ß         | ß         | ß         | £         |
| 253 | FD  | à         | à         | à         | £         |
| 254 | FE  | á         | á         | á         | £         |
| 255 | FF  | â         | â         | â         | £         |



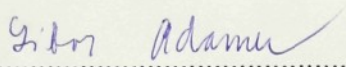
## **Prohlášení k využívání výsledků diplomové práce**

Jsem si vědom, že diplomová práce je majetkem školy a že s ní nemohu sám bez svolení školy disponovat, že diplomová práce může být zapůjčena či objednána (kopie) za účelem využití jejího obsahu.

Beru na vědomí, že po 5ti letech si mohu diplomovou práci vyžádat v Univerzitní knihovně TU v Liberci, kde je uložena.

Jméno a příjmení (rodné) : Libor Adamec

Adresa : Pertoltická 151, MIMOŇ 47124

Podpis : .....  .....