Středoškolská technika 2010

Setkání a prezentace prací středoškolských studentů na ČVUT

Zpracování dat z lineárního lisu

Ondřej Maslikiewicz

SPŠ Hronov
Hostovského 910, Hronov

Úvod

Tento program vznikl za účelem zpracovávání a grafického vyhodnocení naměřených dat na zkušebním stroji INSPEKT 250. Jedná se o hydraulický lis s maximální sílou 250 kN. Pro tento lis byly navrženy speciální přípravky, ve kterých jsou zkoušeny rheologické vlastnosti olejnatých semen. Naměřená data z jednotlivých měření jsou zaznamenávána do tabulek o různém počtu řádků. Počet řádků se liší řádově až v desítkách tisíc. K tomuto účelu se obvykle hodí jakýkoliv tabulkový procesor (Excel, Calc). V tomto případě je však použití tohoto typického způsobu zpracování získaných hodnot nevyhovující. Každý si dokáže představit práci s relativně malou tabulkou, čím však bude tabulka větší, tím se stává méně přehlednou. Už i tabulka větší než je jedna stránka může dělat někomu problémy. Ano, tabulkový procesor za nás téměř všechny hledané a požadované hodnoty najde „sám“, ale někdo mu musí říci, co má hledat. A pracovat s tabulkou, která má přes 50 000 řádků, není nic jednoduchého. Toto obrovské množství dat je výstup měřících čidel na lineárním lisu. Program nemá za úkol nahradit tabulkový procesor. Jedná se o jednoúčelový nástroj pro zpracování pouze dat z měření na lisu.

Celý program je napsán v C++ ve vývojovém prostředí C++ Builder od firmy Borland. V tomto vývojovém prostředí se učíme na škole programovat, proto jsem si ho zvolil.

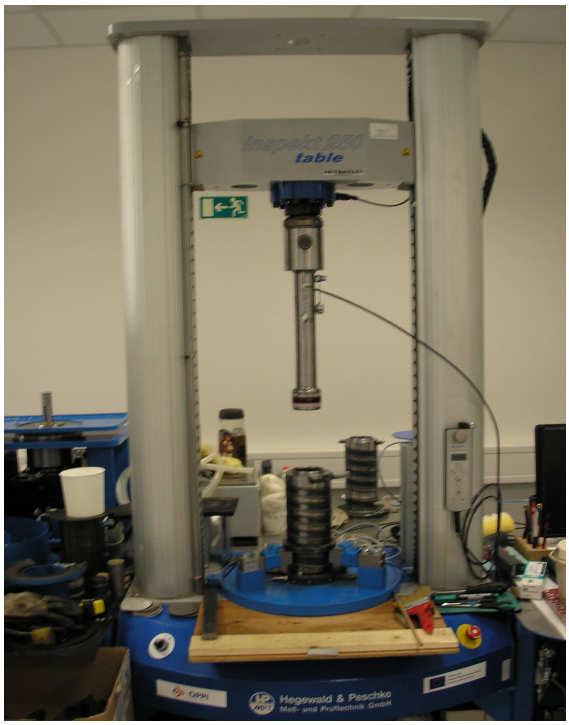
Lineární lis

Lineární lis Inspekt 250 slouží ke stlačování a pohánění lineárních, rotačních a protlačovacích přípravků. Přípravky byly navrženy pro zjišťování rheologických vlastností lisovaných materiálů. Tyto poznatky jsou důležité pro optimalizaci návrhů šnekových geometrií lisů olejnatých semen.

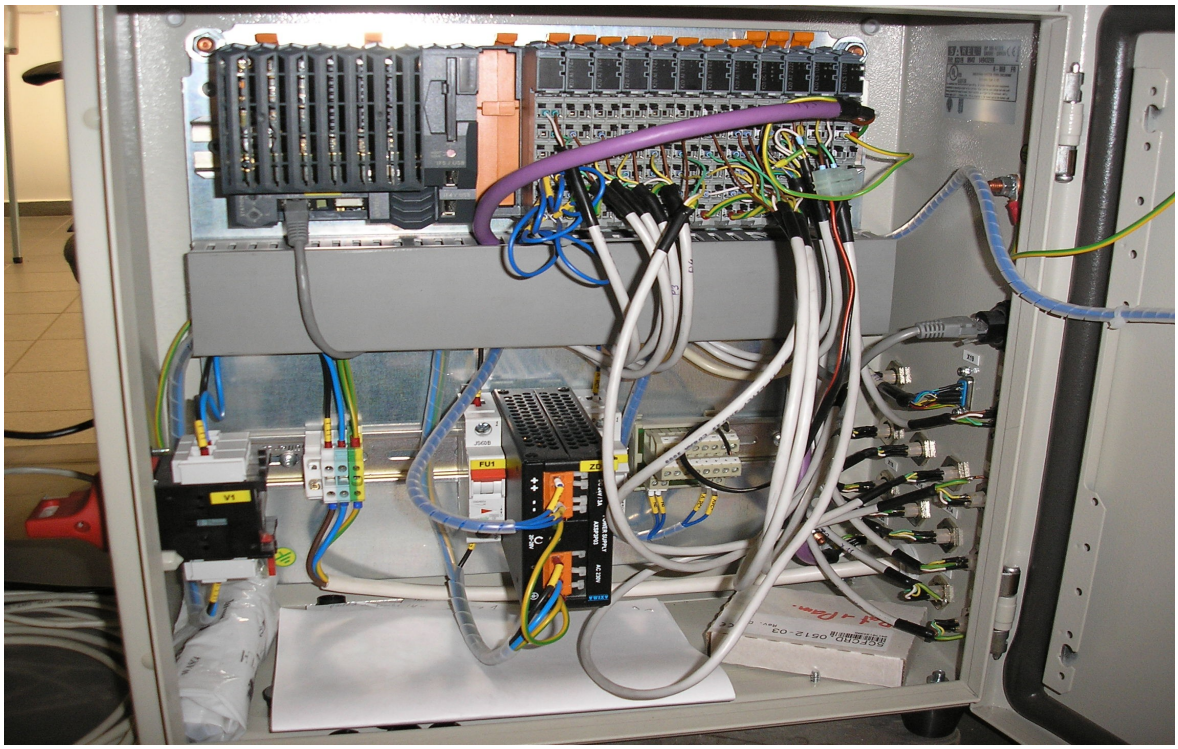
V tomto lineárním přípravku lze měřit:

- hustotu komprimátu
- olejový bod
- stlačitelnost PL matrice
- intenzitu výtoku expulzátu při izobarickém stlačení
- intenzitu výtoku expulzátu při definovaném profilu stlačení
- měření permeability při externím dávkování oleje dutým pístem
- měření pevnosti v 3D experimentu (viz Helwany)
- měření stlačitelnosti komprimátu
- měření difůzních parametrů
- měření výše uvedených vlastností v závislosti na teplotě (přípravek je možné vyhřívat)

Lineární přípravek se skládá z válce, z komory s propustným dnem a z pístu s těsněním, který se zasouvá do válce. Pohyb pístu je zajištěn strojem Inspek 250. Tento stroj zajišťuje lineární pohyb pístu danou rychlostí s omezením maximální síly. Válec je umístěn na tenzometrech, které slouží pro měření třecí síly pístu ve válci. Přes propustné dno válce je pomocí podtlaku odsáván vytlačený olej, který je vážen.



Obr. 1. Lineární lis



Obr. 2. Vnitřní elektrické zapojení měřícího zařízení (PLC Automat)

Vstupní data

Jako vstup pro samotný program slouží csv soubor. Soubor je přímo generován měřící soustavou a je jednoduché s ním pracovat. Tato možnost nahrání dat nebyla jediná. Nabízely se hned tři možnosti. První možností je nahrávání z textového souboru txt. Tato možnost je v našem případě nevýhodná vzhledem k tomu, že pracujeme s daty v tabulce. Nahrávání textového souboru by znamenalo zbytečné zdržení při třídění dat do sloupců. Další už přijatelnější možnost byla nahrávat data ze souborového systému xls, což je primární typ souboru pro tabulkový procesor Excel. Tento postup je poněkud složitější a časově náročný. Třetí možností je již zmiňované načítání z csv souboru. Všechny možnosti byly vyzkoušeny a změřena časová náročnost. Nejlépe, co se týče náročnosti na výpočetní výkon a jednoduchosti naprogramování, vyšlo použité řešení - nahrání ze souboru csv.

Stručně o C++

Programovací jazyk C++ vychází z jazyka C, autory jehož první verze byli Brian W.Kernighan a Denis M. Ritchie. Tato verze označována podle podle příjmení autorů K&R se stala dlouho dobu standardem jazyka C.

Jazyk C++ na rozdíl od jazyka C obsahuje prostředky pro objektově-orientované programování. Většinu zdrojových kódů napsaných v jazyce C lze přeložit překladačem C++. [1] str 34

Vývojové prostředí C++ Builder

C++Builder je vývojový nástroj, který umožňuje navrhovat a vytvářet aplikace pod operačním systémem Windows. Poslední verze C++Builderu fungují pod všemi nejrozšířenějšími verzemi Windows. Na trhu se objevil v roce 1997 a neustále se vyvíjí, poslední verze je z roku 2009.
Prostředí C++Builderu je, jak název napovídá, založeno na jazyce C++. Výkonný zdrojový kód aplikace je zapsán v C++, v prostředí C++Builder navrhujeme především vizuální vzhled aplikace.
Programování v C++Builderu je z velké části založeno na použití komponent. Komponenta je malý program (balíček funkcí), který vykonává určitou činnost (například zobrazuje text nebo obrázky, přehrává multimédia,…).
Velkou předností C++Builderu jsou knihovny komponent, které jsou jejich součástí (např. VCL, CLX, …). Dodávané komponenty významně usnadňují tvorbu aplikací. Další komponenty lze stáhnout z internetu (některé jsou zadarmo, některé se musí koupit). V C++Builderu lze vytvářet vlastní komponenty.
C++Builder je vydáván v různých verzích s rozdílnými znaky a odlišnou cenou: Personal, Professional, Enterprise a Architekt (seřazeno od nejlevnější po nejdražší).

Charakteristické znaky vývojového prostředí C++Builder

- využívá VCL (Visual Component Library) a CLX (Component Library for Cross Platform)
- možnost propojení s databázemi
- tvorba a použití komponent (resp. možnost importu existujících komponent např. z webu
- používání vlastních zpráv k vyvolávání událostí jednotlivých tříd
- objektový model je nezávislý na počtu implementací jednotlivých tříd
- možnost kompilace do x86 kódu

Výhody

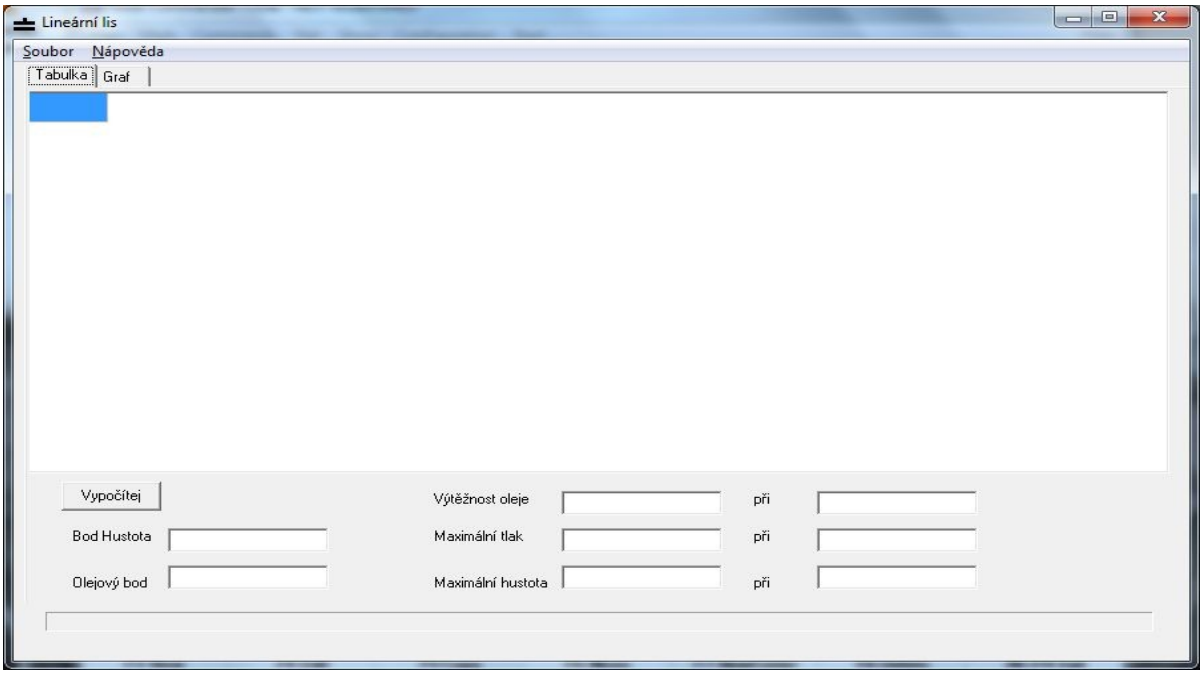
- podpora systému RAD (Rapid Application Development)
- založení na vyšším programovacím jazyce
- možnost kompilace do jednoduchého spustitelného kódu s eliminací funkcí dynamických knihoven
- podpora VCL (Visual Component Library), importu komponent a nástrojů (dokumentace, ladění atd.)
- rychlá optimalizace kódu pro převedení do jazyka symbolických adres
- kompatibilita zdrojových kódů vytvořených ve starších verzích vývojového prostředí s novými verzemi.
- znaky objektově orientovaného programovacího jazyka s možností dědičnosti a polymorfismu v rámci objektových tříd

Nevýhody

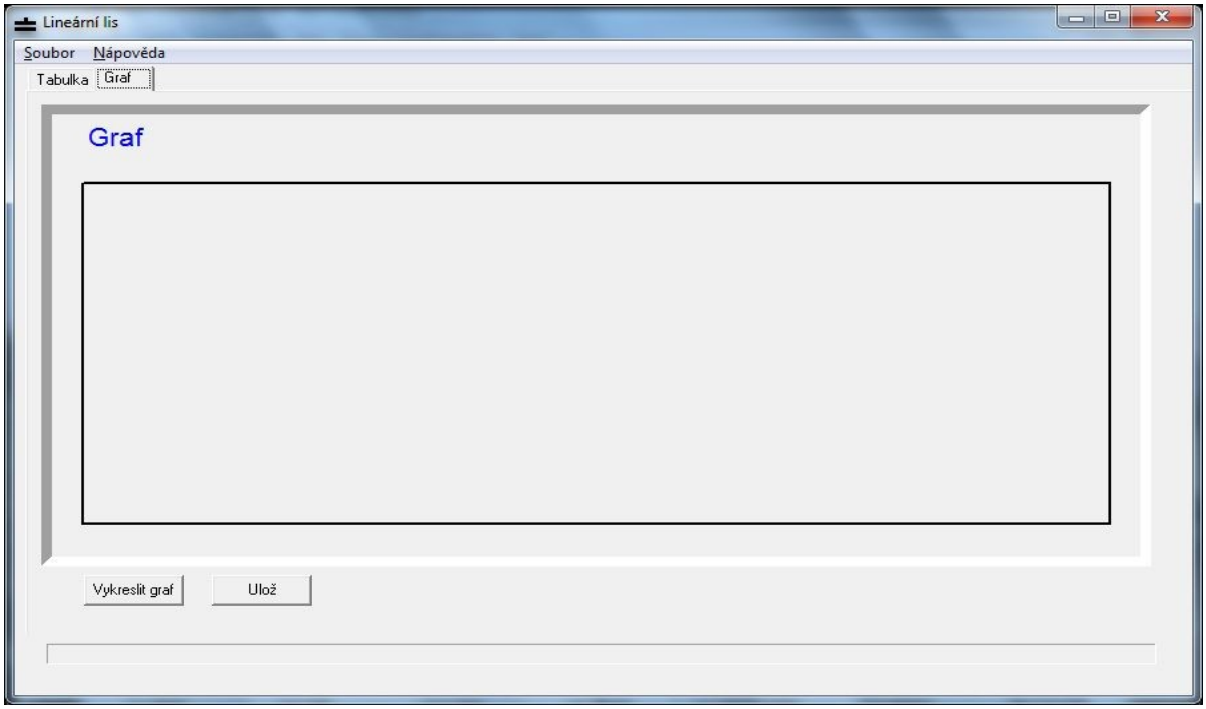
Nevýhodou je, že vytvořené aplikace mohou běžet jen pod OS Microsoft Windows.

Vzhled aplikace

Po spuštění aplikace se otevře okno, na kterém jsou dvě záložky. Na první záložce je komponenta, která slouží k zobrazení tabulky *(obr. 3)*. Komponentou rozumíme to, co se jako komponenta nazývá ve vývojovém prostředí C++ Builder. Dále na této záložce je několik editačních polí, která jsou určena pouze pro čtení. Na těchto „editech“ se zobrazují výsledky. Na druhé záložce je komponenta pro vykreslení a zobrazení grafu *(obr. 4)*. Dále se na obou záložkách nacházejí tlačítka pro ovládání.



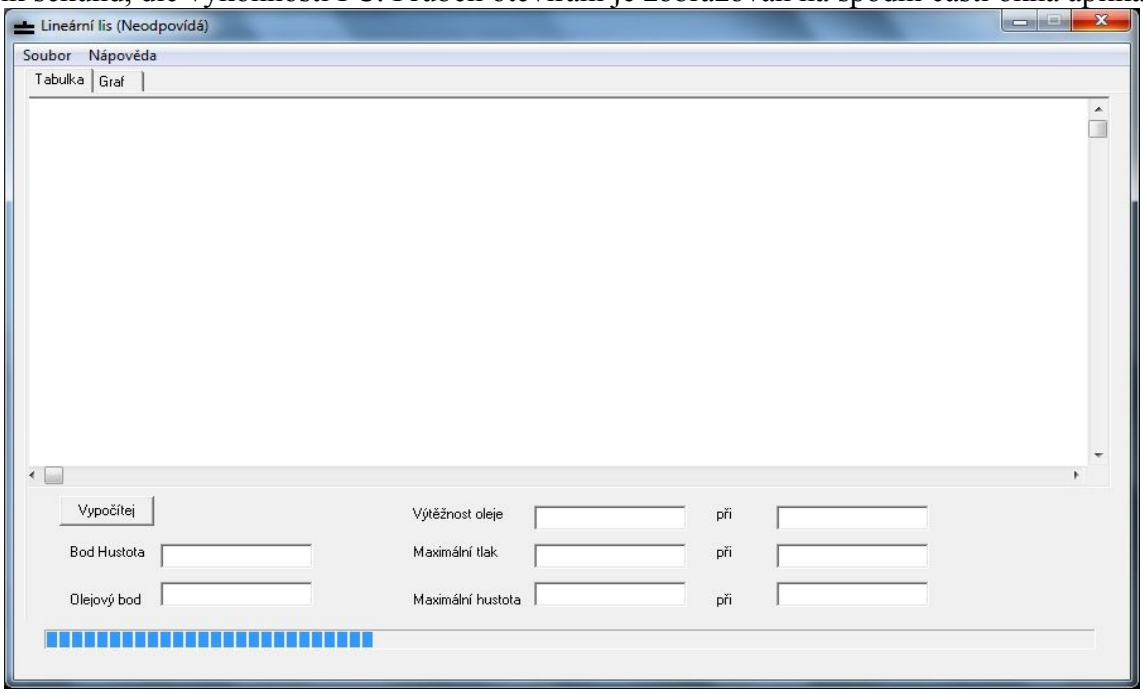
Obr. 3. Vzhled po spuštění 1. záložka



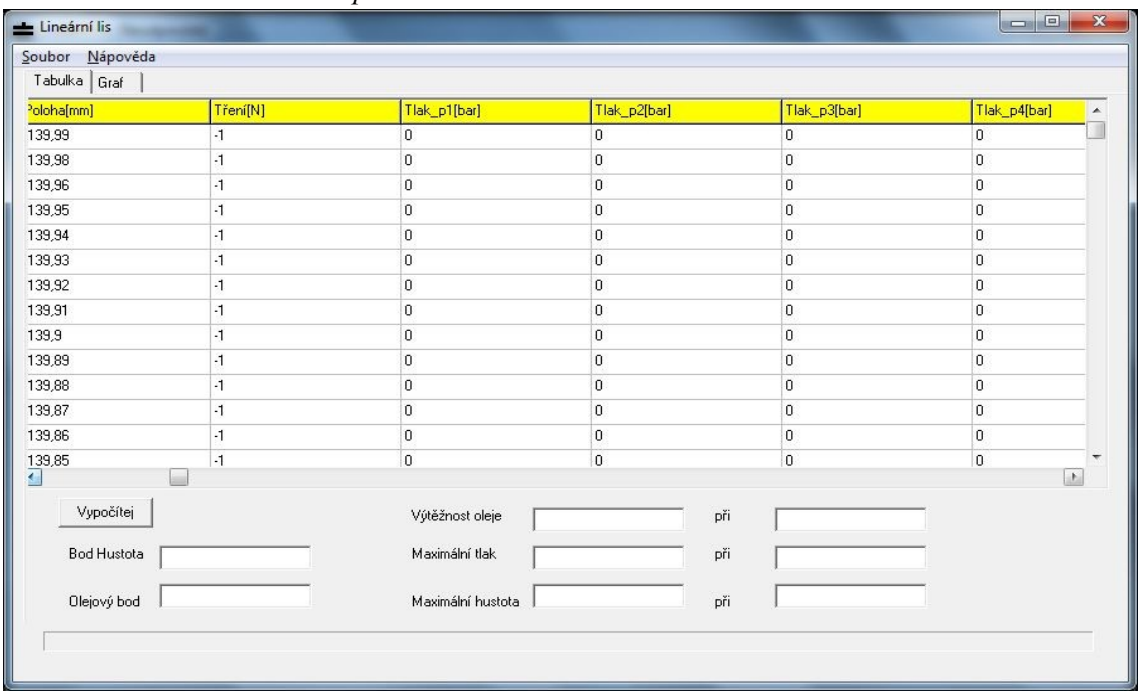
Obr. 4. Vzhled po spuštění 2. záložka

Ovládání a funkce aplikace

Nejprve musíme nahrát soubor do aplikace. V menu soubor vybereme možnost *Otevřít*. Otevře se nám dialogové okno, které je součástí operačního systému a známe je z jiných programů. Pomocí tohoto *OpenDialogu* si vybereme daný soubor. Pro výběr je „od začátku “ nastavené zobrazování csv souborů. V tomto souboru musím být uložena tabulka s naměřenými hodnotami a také musí obsahovat potřebné sloupce jako má vzorový soubor. Sloupce mohou být v libovolném pořadí, ale musí mít stejnou hlavičku jako má soubor„lis.csv“. Po potvrzení otevření se obsah souboru nahraje do komponenty na první záložce. Vzhledem k rozsahu tabulky tato operace trvá několik sekund, dle výkonnosti PC. Průběh otevírání je zobrazován na spodní části okna aplikace.

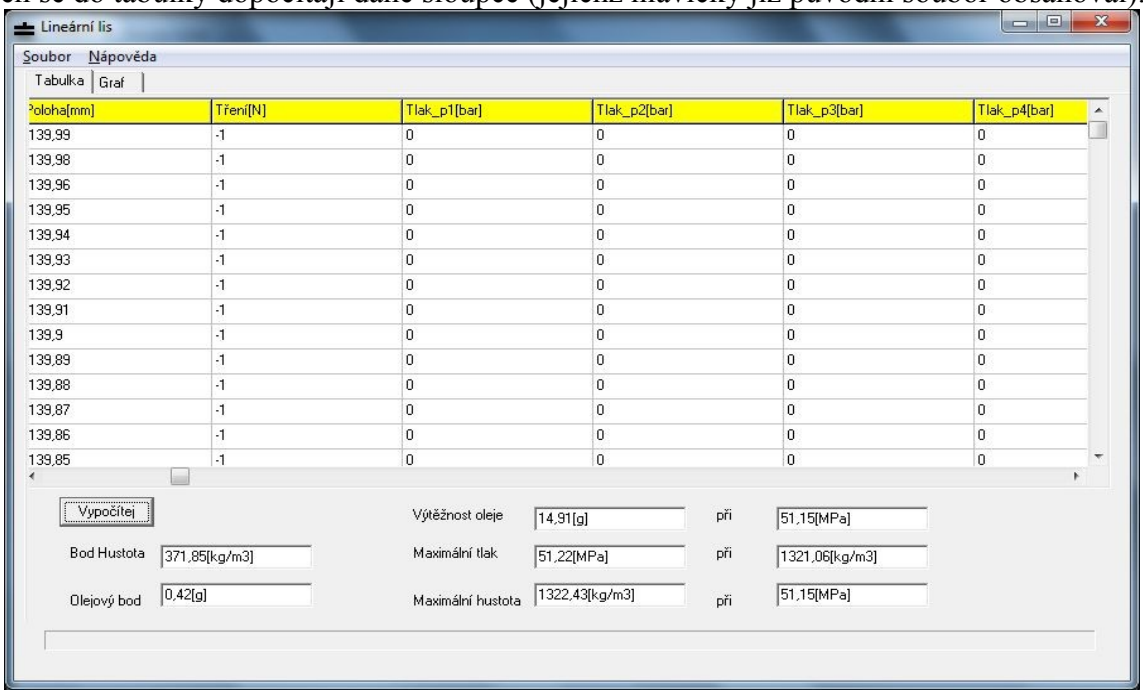


Obr. 5. Nahrávání souboru do aplikace



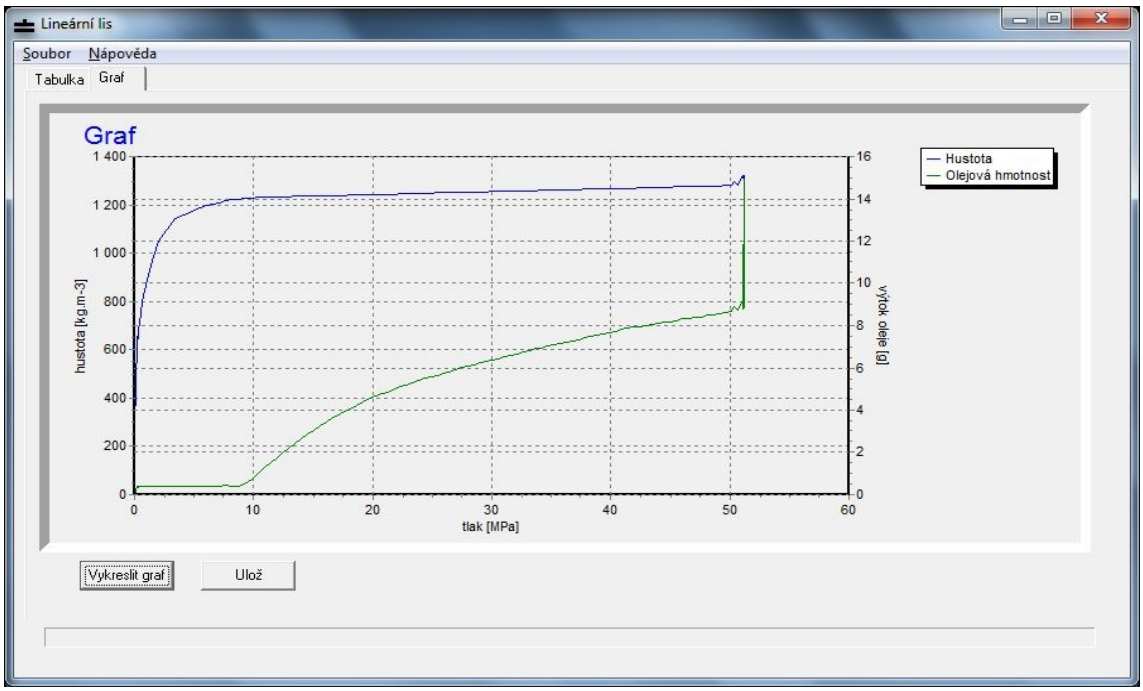
Obr. 6. Zobrazení nahraného souboru v tabulce

Po načtení dat můžeme stisknout tlačítko *Vypočítej* pro výpočet daných sloupců a zároveň pro nalezení určitých bodů. Průběh tohoto procesu se již nezobrazuje. K tomu, proč není zobrazován průběh výpočtu, se vrátím při popisu samotného programu a jeho funkce v *Popis zdrojového kódu (str.)*. Po dokončení výpočtu se v editačních polích zobrazí nalezené body a zároveň se do tabulky dopočítají dané sloupce (jejichž hlavičky již původní soubor obsahoval).

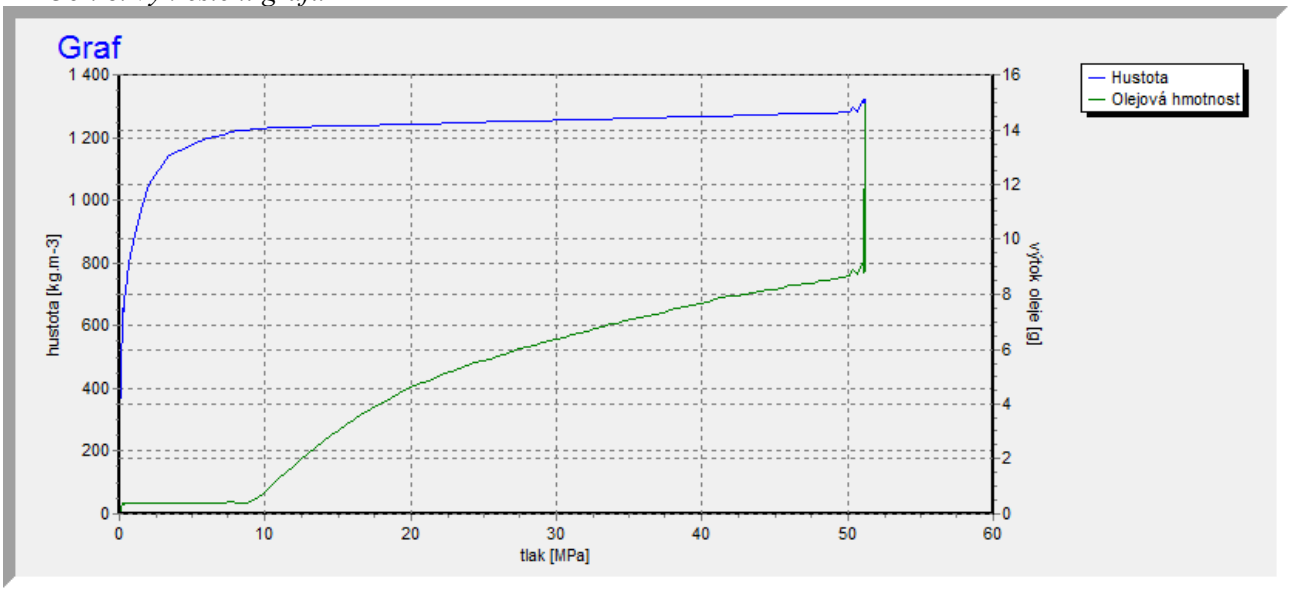


Obr. 7. Zobrazení nalezených hodnot

Z načtených a spočítaných hodnot lze na druhé záložce nakreslit graf *(obr. 8)*. Pro toto slouží tlačítko *Vykreslit graf*. Vykreslený graf lze uložit. Uložení lze provést dvěma způsoby. Buď můžeme použít tlačítko *Uložit*, potom se graf uloží automaticky do stejné složky, ve které se nachází otevřený csv. Soubor. Graf bude uložen ve formátu bitmapy bmp a jeho jméno bude *Graf.bmp (obr. 9)*. To samé má za výsledek kliknutí na možnost *Uložit* v menu *Soubor*. Druhou možností uložení je v menu *Soubor* kliknout na *Uložit jako*. Již název napovídá, že jde o uložení s možností výběru adresáře i jména výsledného souboru. Typ souboru je již přednastaven na bitmapu. Bitmapa jako výstup byla vybrána pro jednoduchost zpracování v programu.



Obr. 8. Vykreslení grafu



Obr. 9. Výsledný soubor Graf.bmp

Dále se v menu *Soubor* vyskytuje možnost ukončení celé aplikace. Použití tohoto podmenu je shodné s kliknutím na křížek pro zavření aplikace. Program se nás před samotným zobrazením zeptá, zda chceme výsledný graf uložit. Pokud jsme graf uložili před ukončením aplikace, tento dialog se nezobrazí a program bude rovnou ukončen.

Popis zdrojového kódu

Na začátku zdrojového kódu je obsluha události *FormCreate*. Tato událost nastane po spuštění aplikace. Jde o vytvoření samotného formuláře, což je komponenta vývojového prostředí C++ Builder. Ve své podstatě jde o okno aplikace. V obsluze události *FormCreate* je tedy napsán samotný vzhled aplikace. Nenachází se zde však úplně všechno. Pozice a některé vlastnosti komponent umístěných na formuláři nejsou napsány ve zdrojovém kódu, ale stará se o ně samotné vývojové prostření. Vzhled byl vytvořen grafickým programováním, což zjednodušuje a zrychluje programování. Právě díky možnosti komponenty rozmístit grafickým programováním máme možnost vytvářet vzhled přímo. Do obsluhy události *FormCreate* se pak pouze napíší příkazy, které vzhled upravují (přímo jej nevytváříme). Pro příklad - nastavujeme zde popisy jednotlivých komponent, jejich barvu, popřípadě přesnou velikost atd. Většinu těchto vlastností lze nastavit přímo v *Buildru*. Tato možnost má však nevýhodu v tom, že po otevření pouze zdrojového kódu nemáme žádnou představu, jak aplikace vypadá.

Následuje obsluha události kliknutí na menu *pmnOtevrit*. Nejprve před zahájením samotného otvírání, respektive načítání, souboru je kontrola, zda-li bylo otevření souboru potvrzeno. Potvrzení je v *OpenDialogu* buď tlačítkem *Otevřít*, nebo *Entrem*, tak jako jsme zvyklí z jiných aplikací. *OpenDialog* patří ke standardním dialogům (dialogovým oknům). Tyto dialogy jsou součástí operačního systému, proto je známe i z jiných aplikací. Kontrola potvrzení je přes jednu z vlastností dialogu a to konkrétně vlastnost *Execute*. Tato vlastnost nabývá dvou hodnot true/false. Díky tomu je celá kontrola provedena podmínkou *if*. Je-li podmínka splněna, tzn. otevření bylo potvrzeno, provede se načtení souboru do komponenty *StringGrid* (tabulka, mřížka). Cesta k souboru je uložena ve vlastnosti OpenDialogu *FileName*. Soubor se nejprve načte do paměti a teprve poté je nahrán do tabulky. Toto řešení umožňuje zobrazovat průběh načítání. Načtení do paměti je rychlé a před načítáním do tabulky již známe velikost souboru. Samotné načtení do tabulky je v cyklu *for*. Zároveň v tomto cyklu probíhá zobrazení průběhu na komponentě *ProgressBar*. Celá tato událost je ošetřena bloky *try* a *catch.* V bloku *try* je napsána obsluha, neboli to, co chceme hlídat, zda nedošlo k chybě. A v bloku *catch* jsou příkazy, které se mají provést při chybě v bloku *try*. V případě tohoto programu je v bloku *catch* pouze jeden příkaz, a to zobrazení chybové hlášky. Tato hláška je vytvořena *API* funkcí a to *MessageBox*. *API*(Application Programming Interface) funkce jsou funkce, které obsahuje operační systém, tzn. dají se použít v libovolném vývojovém prostředí a známe je i z jiných aplikací stejně jako dialogy.

Po obsluze otevření souboru následuje ve zdrojovém kódu obsluha kliknutí na tlačítko *btnPocitej*. Tato obsluha má stejně jako předešlá obsluha bloky *try/catch* pro ošetření chyb při výpočtu. V bloku *try* je několik funkcí nejen pro výpočet, ale i pro hledání určitých bodů v datech naměřených na lineárním lisu. Všechny funkce nemají žádné vstupní, ani výstupní parametry. Žádná z funkcí se ani neopakuje nikde jinde v programu. Z tohoto důvodu by se celá obsluha dala napsat bez použití funkcí. Funkce jsou ovšem použity pro přehlednost zdrojového kódu. Napsat celou obsluhu události kliknutí na tlačítko do jednoho bloku příkazu by bylo velmi nepřehledné. Celá obsluha je opět ošetřena bloky *try/catch*.

Funkce *sloupce* slouží pro určení pořadí sloupců podle jejich hlavičky. Tímto je zaručena nezávislost na pořadí sloupců v souboru (některé nepotřebné sloupce nemusí soubor vůbec obsahovat). Samotnou funkci nalezneme na konci zdrojového kódu. Každý sloupec v tabulce, respektive v komponentě *StringGrid* je určen číslem od nuly. Dané sloupce v souboru jsou vyhledávány pomocí porovnání textu v prvním řádku (hlavičce řádku) a cyklu, který postupně projde všechny buňky na prvním (nultém) řádku. Index správného řádku je uložen do proměnné, která je deklarovaná na začátku zdrojového kódu.

Následuje funkce *Vobjem*. Tato funkce spočítá hodnoty objemu a zapíše je do předem připraveného sloupce.

$$V = \frac{\pi \cdot 1225 \cdot Poloha \cdot (-1)}{10^9} [m^3]$$

Funkce obsahuje cyklus *for* pro výpočet ve všech řádcích tabulky. Následují funkce *Vhmotnost*_vzorku, *Vhustota* a *Vtlak*. Tyto funkce jsou podobné funkci *Vobjem* a jak jejich název napovídá, slouží k výpočtu:
Hmotnosti vzorku:

$$m_{vzorek} = 200 - Olej_{hmotnost} [g]$$

Hustoty:

$$\rho = \frac{m_{vzorek} \cdot V}{1000} [kg\, m^{-3}]$$

Tlaku:

$$p = \frac{F - F_t}{Obsah\ plochy} \cdot 10^{-6} [MPa]$$

Dále následují funkce pro nalezení určitých bodů v tabulce. První z těchto funkcí je funkce *BodHustota*. Tato funkce v tabulce ve sloupci Tření hledá bod, kdy se hodnota začne lišit od -1. Tento bod znamená, že lis začal působit silou. Hledání je uskutečněno pomocí cyklu *while* a podmínky *if*. Jak už bylo řečeno při popisu funkce samotného programu, při počítaní a hledání v tabulce se nezobrazuje průběh. Ten není zobrazován právě kvůli použití cyklu *while*. U tohoto cyklu totiž dopředu neznáme počet opakování, tzn. nemůžeme určit maximální hodnotu průběhu komponenty *ProgressBar*.

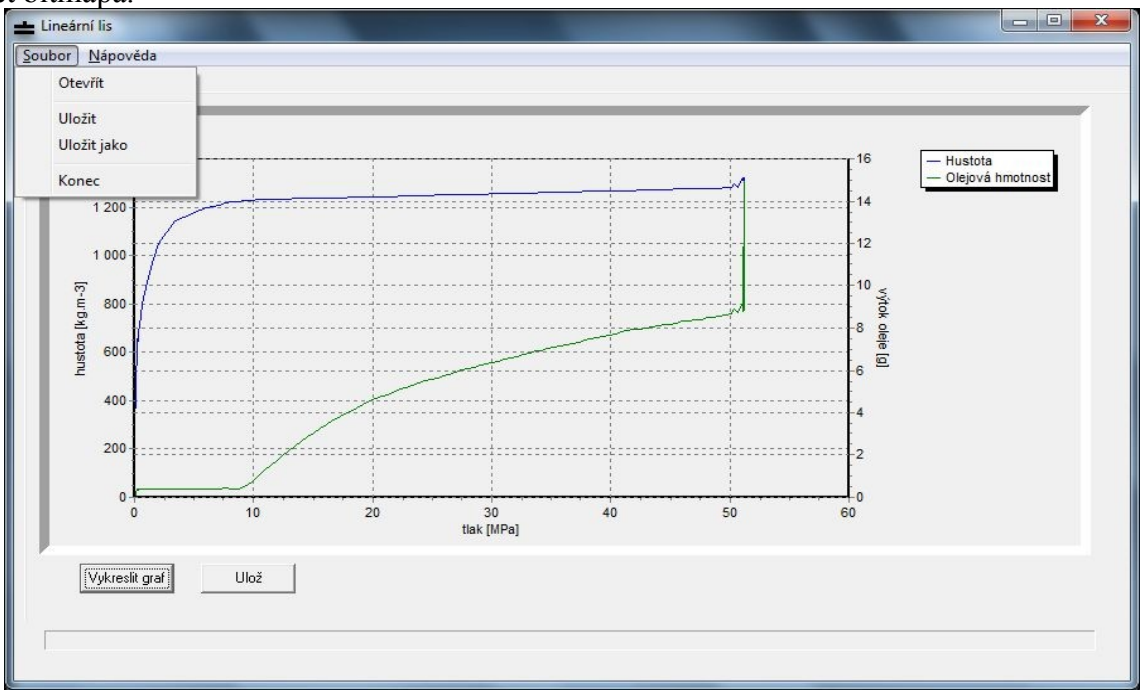
Další funkce pro hledání v tabulce je *OlejBod*, která vyhledá olejový bod. Tento bod je vyhledáván ve sloupci Olej hmotnost, což je hmotnost vylisovaného oleje. Hodnota v tomto sloupci ze začátku osciluje a roste velmi pomalu, úkolem funkce je nalézt místo kde hodnoty přestanou oscilovat a začnou růst. Toto vyhledání je realizováno pomocí cyklu *while*, bloků *try/catch* a podmínky *if*. Hledání je velmi náročné a zatím je řešeno porovnáním třech za sebou následujících hodnot s krokem 100, jelikož se hodnoty několikrát za sebou opakují. Možností zdokonalit tohoto hledání je vzít všechny hodnoty, odstranit opakující se a dále hledat pouze v takto upraveném sloupci. Poté by krok byl jedna a hledání by bylo přesné a nezáleželo by u každého souboru na správné volbě kroku.

Funkce *MaxTlak* a *MaxHustota,* jak jejich název napovídá, vyhledávají maximální tlak a hustotu. Tyto funkce jsou velmi podobné a jsou založeny na nalezení největší hodnoty ve sloupci tabulky. Zároveň s hodnotou maximálním tlakem a hustotou se zobrazuje, při jaké hustotě respektive tlaku toto maximum nastane. Poslední funkce je *VyteznostOlej,* ta ve sloupci Olej hmotnost vyhledává maximální hodnotu.

Dále následuje obsluha události kliknutí na tlačítko *btnGraf*. V této obsluze je napsána část programu pro nakreslení grafu. Stejně jako předešlé obsluhy i tato je ošetřena proti chybě při vykreslování. Samotná obsluha nákresu celého grafu na komponentu … Tato komponenta umožňuje kreslit grafy z datových sérií. Nejprve je nastaven vzhled grafu, přesněji tedy osy. A to jejich popisy a minimální a maximální hodnoty. Graf má zobrazovat dva průběhy a každý má jinou závislou proměnnou, nezávislá proměnná je u obou průběhů stejná a je jí tlak. Z tohoto důvodu je důležité, aby graf obsahoval dvě x-ové osy. Jak je zvykem, jedna je umístěna vpravo a druhá vlevo po okrajích grafu. Poté jsou vytvořeny dvě datové série, každá svázaná s jednou x-ovou osou grafu. Jeden průběh zobrazuje závislost hustoty na tlaku, druhý průběh je závislost olejové hmotnosti taktéž na tlaku. Samotný graf, respektive datové série jsou tvořeny vykreslováním bodu po bodu, kde se vždy udává x-ová a y-ová souřadnice, jak je tomu u kreslení grafu standardní.

Uložení grafu je naprogramováno v obsluze kliknutí na tlačítko *btnUloz*. Při ukládání je použita neviditelná komponenta *Image,* na kterou se celý graf překreslí pomocí kopírování do schránky, tedy do paměti. Výsledný obrázek (graf) se automaticky uloží na adresu, na které se nachází původní otevřený csv soubor. Výsledný obrázek se bude jmenovat *Graf.bmp,* jak přípona

prozrazuje, jde o bitmapu. Dále se v obsluze nastaví boolovská proměnná pro kontrolu uložení. Tato hodnota se dále využívá při ukončení aplikace. Událost kliknutí na položku v menu *pmnUloz* přes *sender* vyvolá obsluhu kliknutí na tlačítko *btnUloz*, nebo-li se provede totéž jako při kliknutí na tlačítko *Ulož*. *Uložit jako* funguje podobně jako *Uložit,* je zde použit stejný princip uložení přes komponentu *Image*. Změna nastává při ukládání na konkrétní místo. Aplikace otevře *SaveDialog*, kde si sami můžeme vybrat, do jakého adresáře a pod jakým jménem bude graf uložen. Typ souboru je opět bitmapa.



Obr. 10. Menu Soubor, vykreslený graf

Poslední důležité části zdrojového kódu jsou obsluhy položky v menu pro ukončení celé aplikace *pmnKonec* a kliknutí na křížek v pravém horním rohu okna aplikace. Před samotným zavřením aplikace se nejprve kontroluje, zda byl výsledný graf uložen, či nikoliv. Pokud nedošlo k uložení grafu ať pomocí *Uložit*, nebo *Uložit jako,* aplikace se zeptá, zda nechceme graf uložit. V opačném případě ,tzn. graf již byl uložen, touto otázkou nebudeme obtěžováni a aplikace se hned vypne.

Poslední část zdrojového kódu je již pro samotnou funkci nepodstatná obsluha událostí kliknutí na položky v menu *Informace o programu* a *Nápověda*. V události *Nápověda* je otevírán jiný formulář, který obsahuje komponentu *Memo,* v které je nápověda pro obsluhu programu napsána. Informace jsou pouze na API funkci *MessageBox*.

Závěr

Program je funkční, zatím slouží jako ukázkový. Při vývoji nových lisovacích technologií je nutno použít právě lineární lis pro jeho exaktnost.
Pro využití programu v praxi je nutno provést změny popisované v kapitole *Popis zdrojového kódu,* aby bylo možno využít program pro libovolný soubor dat se zárukou správnosti stanovení olejového bodu. V našem případě je program využitelný jen pro ukázkový soubor dat. Nejnáročnější bylo správné vyhodnocení dat ze souboru, především hledání olejového bodu.
Práce dokazuje, že C++Builder je vhodný pro programování v praxi, nejen ve školní výuce.

Seznam použité literatury

[1] KADLEC, Václav: Učíme se programovat v C++Builder a jazyce C++, Computer Press, a.s. Brno, 2004

Příloha

Zdrojový kód

```
#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include "Lin_lis.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmLin_lis *frmLin_lis;
#include <memory>
#include <StrUtils.hpp>
#include <Series.hpp>
#include "Napoveda.h"
using namespace std;
//---------------------------------------------------------------------------
__fastcall TfrmLin_lis::TfrmLin_lis(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
  int objem, poloha, hmotnost_vzorku, Olej_hmotnost, hustota, tlak,
    sila, treni, obsah;                  //poromene pro spopecky v tabulce
  bool ulozeno = false;                    //promena pro kontrolu ulozeni
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::FormCreate(TObject *Sender)
//nasteveni vzhledu aplikace

{
  this->Caption = "Lineární lis";
  this->BorderStyle = bsSingle;
  this->Position = poScreenCenter;

  pgcZalosky->Height = this->ClientHeight - pnlProces->Height;
  pgcZalosky->Width = this->ClientWidth;

  btnPocitej->Caption = "Vypoèítej";
  btnPocitej->TabOrder = 0;

  lblHustota->Caption = "Bod Hustota";
  edtHustota->Clear();
  edtHustota->ReadOnly = true;

  lblOlej->Caption = "Olejový bod";
  edtOlej->Clear();
  edtOlej->ReadOnly = true;

  lblMax_tlak->Caption = "Maximální tlak";
```

```
edtMax_tlak->Clear();
edtMax_tlak->ReadOnly = true;

lblMax_hustota->Caption = "Maximální hustota";
edtMax_hustota->Clear();
edtMax_hustota->ReadOnly = true;

lblTlak->Caption = "pøi";
edtTlakpri->Clear();
edtTlakpri->ReadOnly = true;

lblHus->Caption = "pøi";
edtHustotapri->Clear();
edtHustotapri->ReadOnly = true;

lblVyteznost->Caption = "Výtì nost oleje";
edtVyteznost->Clear();
edtVyteznost->ReadOnly = true;

lblVyPri->Caption = "pøi";
edtVyTlak->Clear();
edtVyTlak->ReadOnly = true;

dlgOtevri->DefaultExt = ".csv";
dlgOtevri->Filter = "Soubory csv (*.csv)|*.csv|V echny soubory (*.*)|*.*";
dlgOtevri->Options >> ofHideReadOnly;

dlgUloz->DefaultExt = ".bmp";
dlgUloz->Filter = "Soubory bmp (*.bmp)|*.bmp|V echny soubory (*.*)|*.*";

sgrTab->ColCount = 0;
sgrTab->RowCount = 0;

ChtGraf->Title->Text->Clear();
ChtGraf->Title->Text->Add("Graf");
ChtGraf->Title->Alignment = taLeftJustify;
ChtGraf->Title->Font->Size = 16;
ChtGraf->View3D = false;
ChtGraf->BevelWidth = 8;
ChtGraf->BevelOuter = bvLowered;

btnGraf->Caption = "Vykreslit graf";
btnUloz->Caption = "Ulo ";

pnlProces->Caption = "";
pnlProces->BevelInner = bvNone;
pnlProces->BevelOuter = bvNone;
pnlProces->Width = this->ClientWidth;

imgGraf->Visible = false;
```

```
}
//---------------------------------------------------------------------------
int CountColumns(TStringList *seznam)
{
  int pocet = 0;
  auto_ptr<TStringList> sloupec (new TStringList());
  for (int i = 0; i < seznam->Count; i++)
  {
    sloupec->Delimiter = ';';
    sloupec->DelimitedText = AnsiReplaceText(seznam->Strings[i], " ", "@");
    sloupec->Text = AnsiReplaceText(sloupec->Text, "@", " ");
    pocet = max(pocet, sloupec->Count);
  }
  return pocet;
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnOtevritClick(TObject *Sender)
{
// nacteni souboru
  try
  {
    if (dlgOtevri->Execute())                //byl otevren soubor v dialogu
    {
      auto_ptr<TStringList> seznam (new TStringList());
      seznam->LoadFromFile(dlgOtevri->FileName);

      pbrProces->Min = 0;
      pbrProces->Max = seznam->Count;
      pbrProces->Position = 0;

      sgrTab->ScrollBars = ssBoth;
      sgrTab->DefaultColWidth = 150;
      sgrTab->DefaultRowHeight = 20;
      sgrTab->RowCount = seznam->Count;
      sgrTab->ColCount = CountColumns(seznam.get());
      sgrTab->FixedColor = clYellow;
      sgrTab->FixedCols = 0;
      sgrTab->FixedRows = 1;
      sgrTab->Options = sgrTab->Options << goDrawFocusSelected;
      for (int i = 0; i < seznam->Count; i++)
      {
        pbrProces->Position++;
        sgrTab->Rows[i]->Delimiter = ';';
        sgrTab->Rows[i]->DelimitedText = AnsiReplaceText
          (seznam->Strings[i], " ", "@");
        sgrTab->Rows[i]->Text = AnsiReplaceText
          (sgrTab->Rows[i]->Text, "@", " ");
      }
      if (pbrProces->Position == pbrProces->Max)
        pbrProces->Position = 0;
```

```
    }
   }
   catch(...)
   {
    Application->MessageBox("Nepodaøilo se otevøít soubor!", "Chyba",
    MB_OK + MB_ICONERROR);
   }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::btnPocitejClick(TObject *Sender)
{
  try
  {
  // hledani spravnych sloupcu nezavisle na jejich poradi
    sloupce();

  // vypocty do jednotlivych sloupcu
    Vobjem();
    Vhmotnost_vzorku();
    Vhustota();
    Vtlak();

  // hledani bodu Hustota
    BodHustota();
  // hledani olejoveho bodu
    OlejBod();
  // hledani maximalniho tlaku
    MaxTlak();
  // hledani maximalni hustoty
    MaxHustota();
  // hledani vyteznosti oleje
    VyteznostOlej();
  }
  catch(...)
  {
    Application->MessageBox("Do□lo k chybì ve výpoètu!", "Chyba",
      MB_OK + MB_ICONERROR);
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::btnGrafClick(TObject *Sender)
// vykresleni grafu
{
  try
  {
    ChtGraf->BottomAxis->Automatic = false;
    ChtGraf->BottomAxis->Title->Caption = "tlak [MPa]";
    ChtGraf->BottomAxis->Maximum = 60.0;
    ChtGraf->BottomAxis->Increment = 10.0;
```

```
    ChtGraf->LeftAxis->Automatic = false;
    ChtGraf->LeftAxis->Title->Caption = "hustota [kg.m-3]";
    ChtGraf->LeftAxis->Maximum = 1400.0;
    ChtGraf->LeftAxis->Increment = 200.0;

    ChtGraf->RightAxis->Automatic = false;
    ChtGraf->RightAxis->Title->Caption = "výtok oleje [g]";
    ChtGraf->RightAxis->Maximum = 16.0;
    ChtGraf->RightAxis->Increment = 2.0;

    ChtGraf->SeriesList->Clear();
    TFastLineSeries *data1 = new TFastLineSeries(ChtGraf);
    ChtGraf->AddSeries(data1);
    data1->Title = "Hustota";
    data1->SeriesColor = clBlue;
    for (int i = 1; i < sgrTab->RowCount - 1; i = i + 100)
    {
      data1->AddXY(StrToFloat(sgrTab->Cells[tlak][i]),
        StrToFloat(sgrTab->Cells[hustota][i]), "");
    }

    TFastLineSeries *data2 = new TFastLineSeries(ChtGraf);
    ChtGraf->AddSeries(data2);
    data2->VertAxis = aRightAxis;
    data2->Title = "Olejová hmotnost";
    data2->SeriesColor = clGreen;
    for (int i = 1; i < sgrTab->RowCount - 1; i = i + 100)
    {
      data2->AddXY(StrToFloat(sgrTab->Cells[tlak][i]),
        StrToFloat(sgrTab->Cells[Olej_hmotnost][i]), "");
    }
    ulozeno = false;
  }
  catch(...)
  {
    Application->MessageBox("Nepodaøilo nakreslit graf!", "Chyba",
      MB_OK + MB_ICONERROR);
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::btnUlozClick(TObject *Sender)
{
// ulozeni grafu
  try
  {
    ChtGraf->CopyToClipboardBitmap();
    imgGraf->Picture->LoadFromClipboardFormat
      (CF_BITMAP, Clipboard()->GetAsHandle(CF_BITMAP), 0);
    imgGraf->Picture->SaveToFile(ExtractFilePath(dlgOtevri->FileName) +
      "Graf.bmp");
```

```
    ulozeno = true;
    Application->MessageBox(
      "Graf byl ulo□en jako Graf.bmp ve slo□ce s csv souborem", "Informace",
       MB_OK + MB_ICONINFORMATION);
   }
   catch(...)
   {
    Application->MessageBox("Nepodaøilo se ulo□it graf!", "Chyba",
      MB_OK + MB_ICONERROR);
   }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnUlozClick(TObject *Sender)
{
  btnUlozClick(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnUlozJakoClick(TObject *Sender)
{
// ulozeni grafu jako
  try
  {
    if (dlgUloz->Execute())
    {
      ChtGraf->CopyToClipboardBitmap();
      imgGraf->Picture->LoadFromClipboardFormat
        (CF_BITMAP, Clipboard()->GetAsHandle(CF_BITMAP), 0);
      imgGraf->Picture->SaveToFile(dlgUloz->FileName);
    }
    ulozeno = true;
    Application->MessageBox("Graf byl ulo□en", "Informace",
       MB_OK + MB_ICONINFORMATION);
  }
  catch(...)
  {
    Application->MessageBox("Nepodaøilo se ulo□it graf!", "Chyba",
      MB_OK + MB_ICONERROR);
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnKonecClick(TObject *Sender)
{
// ukonceni aplikace
  if(!ulozeno)
  {
    if (MessageDlg("Chcete ulo□it graf ?", mtConfirmation,
        TMsgDlgButtons() << mbNo << mbYes, 0) == mrYes)
      btnUlozClick(Sender);
  }
  Application->Terminate();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::FormClose(TObject *Sender,
     TCloseAction &Action)
{
 pmnKonecClick(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnOprgClick(TObject *Sender)
{
// info o programu
 Application->MessageBox("Ondøej Maslikiewicz, o.maslik@gmail.com",
     "O programu", MB_OK);
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::pmnNapovedaClick(TObject *Sender)
{
// napoveda k programu
 frmNapoveda->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::sloupce()
{
 for (int j = 0; j < sgrTab->ColCount; j++)
 {
  if (sgrTab->Cells[j][0] == "objem[m3]")
    objem = j;
 }

 for (int j = 0; j < sgrTab->ColCount; j++)
 {
  if (sgrTab->Cells[j][0] == "Poloha[mm]")
    poloha = j;
 }

 for (int j = 0; j < sgrTab->ColCount; j++)
 {
  if (sgrTab->Cells[j][0] == "hmotnost vzorku[g]")
    hmotnost_vzorku = j;
 }

 for (int j = 0; j < sgrTab->ColCount; j++)
 {
  if (sgrTab->Cells[j][0] == "Olej_hmotnost[g]")
    Olej_hmotnost = j;
 }

 for (int j = 0; j < sgrTab->ColCount; j++)
 {
  if (sgrTab->Cells[j][0] == "hustota[kg.m-3]")
```

```
    hustota = j;
  }

  for (int j = 0; j < sgrTab->ColCount; j++)
  {
   if (sgrTab->Cells[j][0] == "Síla[N]")
     sila = j;
  }

  for (int j = 0; j < sgrTab->ColCount; j++)
  {
   if (sgrTab->Cells[j][0] == "Tøení[N]")
     treni = j;
  }

  for (int j = 0; j < sgrTab->ColCount; j++)
  {
   if (sgrTab->Cells[j][0] == "tlak[MPa]")
     tlak = j;
  }

  for (int j = 0; j < sgrTab->ColCount; j++)
  {
   if (sgrTab->Cells[j][0] == "obsah plochy[m2]")
     obsah = j;
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::Vobjem()
{
  for (int i = 1; i < sgrTab->RowCount - 1; i++)
    sgrTab->Cells[objem][i] = FloatToStr
        (-(StrToFloat(sgrTab->Cells[poloha][i])* 1225 * M_PI) / 1000000000);
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::Vhmotnost_vzorku()
{
  for (int i = 1; i < sgrTab->RowCount - 1; i++)
    sgrTab->Cells[hmotnost_vzorku][i] =
       FloatToStr(200 - StrToFloat(sgrTab->Cells[Olej_hmotnost][i]));
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::Vhustota()
{
  for (int i = 1; i < sgrTab->RowCount - 1; i++)
    sgrTab->Cells[hustota][i] =
       FloatToStr(StrToFloat(sgrTab->Cells[hmotnost_vzorku][i]) / 1000 /
       StrToFloat(sgrTab->Cells[objem][i]));
}
//---------------------------------------------------------------------------
```

```
void __fastcall TfrmLin_lis::Vtlak()
{
  for (int i = 1; i < sgrTab->RowCount - 1; i++)
    sgrTab->Cells[tlak][i] = FloatToStr(((StrToFloat(sgrTab->Cells[sila][i]) -
        StrToFloat(sgrTab->Cells[treni][i])) /
        StrToFloat(sgrTab->Cells[obsah][1])) / 1000000);
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::BodHustota()
{
  bool konec = false;
  int i = 1;
  while (!konec)
  {
    if (StrToFloat(sgrTab->Cells[treni][i]) == -1)
      i++;
    else
    {
      konec = true;
      edtHustota->Text = FloatToStr((float)(int)(StrToFloat
          (sgrTab->Cells[hustota][i]) * 100) / 100) + "[kg/m3]";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::OlejBod()
{
  bool konec = false;
  int i = 0;
  while (!konec)
  {
    i = i + 100;
    try
    {
      if ((StrToFloat(sgrTab->Cells[Olej_hmotnost][i + 100]) -
          StrToFloat(sgrTab->Cells[Olej_hmotnost][i])) > 0.04)
      {
        if ((StrToFloat(sgrTab->Cells[Olej_hmotnost][i + 200]) -
            StrToFloat(sgrTab->Cells[Olej_hmotnost][i + 100])) > 0.04)
        {
          edtOlej->Text = FloatToStr((float)(int)(StrToFloat
            (sgrTab->Cells[Olej_hmotnost][i])*100)/100) + "[g]";
          konec = true;
        }
      }
    }
    catch (...)
    {
      konec = true;
    }
```

```
  }
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::MaxTlak()
{
  bool konec = false;
  int i = 0;
  float max_tlak = 0, pri_hustota;
  while (!konec)
  {
    i++;
    try
    {
      if (StrToFloat(sgrTab->Cells[tlak][i]) > max_tlak)
      {
        max_tlak = StrToFloat(sgrTab->Cells[tlak][i]);
        pri_hustota = StrToFloat(sgrTab->Cells[hustota][i]);
      }
    }
    catch (...)
    {
      konec = true;
    }
  }
  edtMax_tlak->Text = FloatToStr((float)(int)(max_tlak * 100) /
                        100) + "[MPa]";
  edtTlakpri->Text = FloatToStr((float)(int)(pri_hustota * 100) /
                        100) + "[kg/m3]";
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::MaxHustota()
{
  bool konec = false;
  int i = 0;
  double max_hustota = 0, pri_tlak;
  while (!konec)
  {
    i++;
    try
    {
      if (StrToFloat(sgrTab->Cells[hustota][i]) > max_hustota)
      {
        max_hustota = StrToFloat(sgrTab->Cells[hustota][i]);
        pri_tlak = StrToFloat(sgrTab->Cells[tlak][i]);
      }
    }
    catch (...)
    {
      konec = true;
    }
```

```
  }
  edtMax_hustota->Text = FloatToStr((float)(int)(max_hustota * 100) / 100)
      + "[kg/m3]";
  edtHustotapri->Text = FloatToStr((float)(int)(pri_tlak * 100) / 100)
      + "[MPa]";
}
//---------------------------------------------------------------------------
void __fastcall TfrmLin_lis::VyteznostOlej()
{
  bool konec = false;
  int i = 0;
  double vy_olej = 0, vy_tlak = 0;
  while (!konec)
  {
    i++;
    try
    {
      if (StrToFloat(sgrTab->Cells[Olej_hmotnost][i]) > vy_olej)
      {
        vy_olej = StrToFloat(sgrTab->Cells[Olej_hmotnost][i]);
        vy_tlak = StrToFloat(sgrTab->Cells[tlak][i]);
      }
    }
    catch (...)
    {
      konec = true;
    }
  }
  edtVyteznost->Text = FloatToStr((float)(int)(vy_olej * 100) / 100) + "[g]";
  edtVyTlak->Text = FloatToStr((float)(int)(vy_tlak * 100) / 100) + "[MPa]";
}
```