Ako správne pripraviť protokol

Táto hodina by Vám mala ponúknuť základné informácie ako „správne" vytvoriť program, napísať protokol a pripraviť prezentáciu, ktoré sú nutné pre udelenie zápočtu z chemicko-inžinierskych výpočtov na PC.

Najprv si povieme niekoľko základných vecí o tom, ako by mal vyzerať program (prípadne funkcia), ktorý budete musieť vytvoriť. V nasledujúcich riadkoch uvediem len akési základné body, ktoré je potrebné mať na zreteli pri písaní programu.

Zdrojový kód

Aby bolo možné sa vo väčšom projekte bezproblémovo orientovať, mali by ste dodržiavať niekoľko zásad:

- ak sa Váš projekt skladá z viacerých súborov, mali by byť uložené spolu v jednom adresári
- mená jednotlivých súborov by mali byť jednoznačné a výstižné
- každý program (funkcia) by mali začínať hlavičkou, v ktorej by mali byť informácie o mene súboru, stručný popis programu, informácie o vstupných a výstupných parametroch

```
%%%Funkcia integral
%opis:
%sluzi na vypocet urciteho integralu funkcie funkcia obdlznikovou metodou
%Vstupne parametre:
%a-dolna hranica integralu
%b-horna hranica integralu
%n-pocet deliacich bodov (priamo umerne presnosti)
%Vystupny parameter:
%I-hodnota urciteho integralu na intervale <a,b>

function [I]=integral_O(a,b,n)
x=linspace(a,b,n);
suma=0;
for i=1:(n-1)
  suma=suma+funkcia(x(i));
end
I=(b-a)/n*suma;
```

Názvy premenných

Syntax MATLABu je „case sensitive" to znamená rozlišuje veľké a malé písmena, čo znamená, že **deriv Deriv DERIV** sú tri odlišné premenné.
Aj pri tvorbe názvu premenných existuje niekoľko zásad, ktoré je dobré dodržiavať:

- nie je veľmi vhodné používať podobné identifikátory (matA matAA)

- V žiadnom prípade nepoužívať dva rovnaké identifikátory, líšiace sa len typom písma (determinant DETERMINANT)

- mena premenných by mali byť výstižne a logické (napr. **najvacsi_prvok** a nie **np** alebo **aaa**)

Komentáre v programe

Aj keď to na prvý pohľad nevyzerá, komentáre predstavujú veľmi dôležitú súčasť zdrojového kódu, nakoľko umožňujú sprehľadnenie rozsiahleho kódu.
Komentáre slúžia hlavne na to, aby sa vo Vašich programoch mohol vyznať aj niekto iný, prípadne aj Vy sami, ak sa k nemu dostanete po dlhšom čase.
Komentár by sa mal uviesť pred každou logicky ucelenou časťou kódu a mal by obsahovať len dôležité informácie.

Všetky vyššie uvedené rady je teoreticky možné aplikovať v ľubovoľnom programovacom jazyku. Sú však určité špecifiká, ktorými sa MATLAB líši od iných programovacích jazykov. Jedným z takýchto špecifík je, že MALTAB automaticky vypisuje každú premennú prípadne výraz , ktorý nie je ukončený bodkočiarkou. Preto by ste mali dávať veľký pozor na to, aby sa na obrazovku vypisovali len tie informácie, ktoré sú skutočne nevyhnutné a to hneď z dvoch dôvodov:

- vypisovanie veľkého množstva údajov značne spomaľuje beh programu

- užívateľa, ktorý používa program, často nezaujímajú jednotlivé medzivýpočty, ale len výsledok a preto sledovanie nekonečne dlhého výpisu čísiel môže pôsobiť dosť rušivo

Jedna z častých požiadaviek na program je, aby bol dostatočne rýchly. Vo vašich projektoch by sa nemal objaviť program, ktorý by vyžadoval mnoho výpočtového času. Beztak určenie času, ktorý program potrebuje je veľmi dôležitá informácia. MATLAB má na túto činnosť hneď niekoľko metód, no my si ukážeme len tú najjednoduchšiu:

```
tic

 casovo narocna cast programu

toc

ubehnuty_cas = toc
```

Niekoľko informácii o tom ako by mal vyzerať protokol

Mal by obsahovať základné informácie o zadaní úlohy, napr:

Zadanie:
Mojou úlohou bolo vytvorenie programu, schopného riešiť hlavolam sudoku. Vstupný údaj do programu je matica rozmeru 81x81, v ktorej sú zadané známe a neznáme. Neznáme údaje sú označené hodnotu 0, ostatné hodnotami 1-9.

Ďalším dôležitým údajom, nielen v protokole, ale aj v samotnom postupe tvorby programu by mal byť **teoretický rozbor**, v ktorom je nutné pojednať o spôsobe, akým riešiť zadaný problém. Táto časť je pravdepodobne najdôležitejšia, nakoľko predurčuje, akým smerom sa bude uberať príprava programu. Mala by stručne informovať o základných častiach programu a väzbách medzi nimi.

Teoretický rozbor:
Úloha vylúštenia sukoku spočíva v doplnení políčok, ktoré sú neznáme tak, aby každý riadok a stĺpec obsahoval čísla od 1 po 9 práve jeden krát, podobne každá z 9 hlavných časti hracieho poľa musí obsah uvedené čísla práve jeden krát.
Spôsobov lúštenia sudoku je viacero. Ja som sa rozhodol použiť nasledovný postup: vytvorím si tri funkcie, z ktorých každá bude schopná určiť, či je splnené jedno z pravidiel sudoku. Nasledovne program určí, aké (koľko) rôznych hodnôt je možné vložiť do každého z políčok. V prípade, že je možné vložiť práve jednu hodnotu, program ju vloží. Uvedený postup sa opakuje do tej chvíle, kým existujú políčka, do ktorých je možné umiestniť práve jednu hodnotu.
Vo chvíli, keď už neexistujú takéto políčka, program prejde do druhej fázy. V tejto fáze už program vyhodnotí hracie pole a zistí, v ktorom políčku je možné umiestniť najmenší možný počet hodnôt (väčšionu dve). Program náhodne vyberie jednu z možností. Tým sa môže stať, že opäť vzniknú políčka, do ktorých je možné vložiť práve jednu hodnotu. Takto program postupne pridáva jednotlivé hodnoty a môžu nastať v podstate dve situácie:

- *program vyplní všetky políčka správne (teda program skončil úspešne)*
- *alebo sa program dostane do bodu, v ktorom už nie je možné umiestniť žiadne políčko, to znamená, že pri náhodnom výbere došlo k chybe. Preto je nutné celú túto časť programu zopakovať. Nakoľko sú použité náhodné čísla, nie je možné určiť, kedy chyba nastala no je možné predpokladať, že po určitom počte pokusov program skutočne nájde správne riešenie.*

Ďalšia fáza protokolu by mala pojednávať o samotnom algoritme. Je vhodné vymenovať všetky časti programu, spôsob ako si medzi sebou vymieňajú informácie. V tejto časti by nemal chýbať kompletný opis programu. Zvlášť podrobne by mali byť vysvetlené kľúčové časti zdrojového kódu.

Jadro:
Srdce programu tvoria tri funkcie:
function [validny]=kontrola_riadok(A,Riadok)
function [validny]=kontrola_stlpec(A,Stlpec)
function [validny]=kontrola_stvorec(A,riadok,stlpec)

tieto funkcie majú za úlohu kontrolovať či prvok, ktorý program umiesti do hracieho poľa je v súlade s pravidlami sudoku. Všetky tri funkcie vracajú jednu hodnotu, ktorá určuje, či sa vložením prvku do matice porušili pravidla sudoku alebo nie. V prípade, že sú pravidla porušené, funkcie vrátia hodnotu 1, v opačnom prípade 0.

Všetky tri funkcie sú veľmi podobné, preto na vysvetlenie problematiky použijeme len jednu:

```
function [validny]=kontrola_riadok(A,Riadok)
validny=0;
for i=1:9
indexy=find(A(Riadok,:)==i);
pocet_rovnakych_prvkov=length(indexy);
if pocet_rovnakych_prvkov>1
validny=1;
end
end
```

v uvedenej funkcii sa testuje, či riadok (špecifikovaný parametrom funkcie) je v súlade s pravidlami sudoku, to znamená, že každá z hodnôt 1-9 sa v danom riadku nachádza najviac jeden krát.

Tieto tri funkcie využíva funkcia:

```
[Pocet_Moznosti,Matica_Moznosti]=zisti_moznosti(A);
```

ktorá zistí pre každú neznámu hodnotu hracieho poľa všetky hodnoty, ktoré je možné do daného poľa umiestniť bez toho, aby došlo k porušeniu pravidiel sudoku.

Funkcia vracia dve polia:
Pocet_Moznosti rozmeru 81x81, v ktorej sú uložené počty hodnôt, ktoré je možné do daného políčka umiestniť. Matica_Moznosti rozmeru 81x81, v ktorej sú uložene všetky hodnoty, ktoré je možné umiestniť do hracieho poľa bez porušenia pravidiel.
.
.
.

Takto by ste mali pokračovať v opise celého algoritmu a zvlášť by ste sa mali zamerať na opis kľúčových častí Vášho programu.

V závere práce by bolo veľmi vhodné skomentovať hlavné výhody Vášho programu. Nemali by ste však zabudnúť ani na uvedenie obmedzení, ktoré Váš program má.

Záver:
Mojou úlohou bolo vytvorenie programu na riešenie hlavolamu sudoku. Použil som postup založený na jednoduchej analýze, ktorá v prvej fáze umiestni hodnoty len do polí, ktoré je možné jednoznačne určiť ako správne. V druhej fáze, pri ktorej neexistujú jednoznačné ťahy, sa program pomocou náhodných čísel pokúša postupne hádať jednotlivé hodnoty. Samozrejme, že pri tomto postupe musí zákonite dôjsť k chybám, preto vždy ak sa stane hracie pole nevalidným, program sa vráti do miesta, v ktorom boli všetky hodnoty vložené bez tipovania (na koniec prvej fázy) a celý postup sa opakuje pokým sa nenájde správne riešenie. Program je v podstate schopný vypočítať akékoľvek zadanie. Medzi jeho nevýhody patrí hlavne časová náročnosť (zvlášť pri komplikovanejších zadaniach).

V podobnom duchu ako uvedený protokol by sa mala niesť aj Vaša prezentácia.