ESTAT, statistické programovací prostředí v Turbo Pascalu

Jiří Žváček, Hana Řezanková, VŠE Praha

1. Úvod

Současnou situaci v oblasti statistického softwaru u nás lze charakterizovat dvěma tendencemi:

a) mechanické používání dostupného, většinou nelegálně získaného softwaru s neadekvátní dokumentací (takže se ani často přesně neví, co se vlastně počítá),

b) amatérské programy vlastních metod, bez uživatelsky přijatelného interface a návaznosti na stávající systémy.

Je zřejmé, že se tím dostáváme mimo hlavní proud vývoje výpočetní statistiky a nejenom nejsme schopni konkurence, ale stáváme se stále více závislí na vývoji v zahraničí.

Otázkou je, zda vůbec existuje východisko z dané situace, které by umožnilo spojit síly těch, kteří se pokoušejí o vývoj vlastních metod a případné ovlivnění vývoje.

Východiska jsou podle našeho názoru dokonce dvě:

I. Navázat na legálně získaný produkt, který obsahuje rozumný uživatelský interface a který by měl interface k nějakému známějšímu obecnému programovacímu jazyku, aby bylo možno doplňovat vlastní produkty. Vzhledem k tomu, že většina statistických algoritmů je veřejně publikována v jazyce FORTRAN 77, měl by to být tedy produkt psaný ve FORTRANu.

V tomto směru se přímo nabízí SYSTAT, který má i svou nekomerční verzi MYSTAT a pro který jsou k dispozici texty FORTRANských procedur pro práci s datovou maticí.

II. Vytvořit si vlastní statistické programové prostředí pro tvorbu programů a výuku jejich tvorby. Vzhledem k potřebám výuky by to měl být zřejmě systém v některém Pascalovském jazyce.

Na VŠE jsme se rozhodli pro tuto druhou cestu a jazyk Turbo Pascal, který ve verzi 5.5 již dostatečně splňuje i požadavky na moderní programování (zamítli jsme tedy počítačově efektivnější jazyk C a alternativně postupujeme v jazyce LISP, který však není vhodný pro PC počítače).

Za základ tohoto programového prostředí byl zvolen Conlonův MLIB, který byl nejenom legálně získán, ale představuje podle našeho názoru velmi komplexní a zároveň instruktivní soubor podprogramů v jazyku Turbo Pascal 3.0. Vzhledem k tomu, že nad tímto souborem pracujeme již téměř dva roky a od té doby se ne-

jenom objevily další 3 novelizace jazyka, ale došlo i k podstatným změnám a doplněním, nazýváme tento nový produkt ESTAT.

Se systémem hodláme i nadále pracovat stejně jako v minulosti a jak bylo původním Conlonovým záměrem, tj. poskytujeme jej bezplatně akademickým institucím a jednotlivcům, kteří se alespoň morálně zaváží ke spolupráci. Zdrojové tvary programových jednotek jsou pro tyto spolupracovníky typu public domain a nemohou tedy být předmětem směny. Pokud budou vytvořeny EXE programy, jsou přirozeně majetkem programátora.

Na druhé straně je však nutno upozornit na skutečnost, že dokumentaci systému vzhledem k dynamice vývoje nevěnujeme velkou pozornost, a systém nelze na vyšší úrovni používat bez znalosti nebo přinejmenším dokumentace původního MLIBu, jehož prioritu nechceme žádným způsobem zpochybnit.

Použití ESTATu vidíme zatím zejména v oblasti výuky konstrukce statistických programových paketů. Teprve při práci s konkrétním systémem jsou patrné i specifický statistické problémy a dochází i k hlubšímu pochopení práce ostatních produktů.

Velmi se nám také osvědčilo spojit práci nad programovým prostředím s konstrukcí konkrétních programů. Současní uživatelé již většinou mají k dispozici osobní počítače a na druhé straně se nám již nezdá únosné předávat komerční programové produkty (které jsou bez dokumentace stejně omezeně použitelné). Konstrukce jednoúčelových programů s přijatelným uživatelským interfacem je již dnes podle našeho názoru v ESTATu možná.

2. Struktura ESTATu

V současné době se skládá ESTAT z následujících programových jednotek:

UMajor Hlavní a nezbytná komponenta prostředí, která obsahuje deklaraci používaných datových typů, nastavení barev a video paměti pro nejběžnější typy karet (CGA, EGA, Hercules), základní komponenty pro práci s maticemi, a mnoho pomocných podprogramů,

UMtxIO Základní podprogramy vstupu a výstupu, práce s obrazovkou, procedury pro editování matice.

Ugreat Jednotka sdružující hlavní komponenty pro konstrukci složitějších programů a paketů.

Umatrx Operace s maticemi.

Ufunkc Statistické a matematické funkce.

Umulti Vícerozměrné metody.

Usortm Třídění a řazení.

Utabls Práce s kontingenčními tabulkami.

Ugener Generování náhodných čísel.

Useries Základní procedury pro práci s časovými řadami.

Usegment Segmentová regrese.

Jednotlivé programové jednotky se skládají z procedur a funkcí. Snažili jsme se (prakticky ovšem pouze u nových podprogramů a při zásadnějších úpravách) dodržet následující konvence

- podprogramy mají pokud možno výstižný anglický název,
- funkce mají tvar anglického slova, kde významové prvky jsou odděleny velkými písmeny, zatímco u procedur plní obdobnou roli podtržítko.

Vycházíme z filozofie, že angličtina je jazykem statistiky a že nelze bez komplikací používat češtinu přímo. Z těchto důvodů je

- ovládání založeno na angličtině,
- vysvětlení a delší texty v češtině (z technických důvodů bez háčků a čárek).

3. Základní komponenty

Lze v podstatě říci, že systém je postaven na dvou základních datových typech

MATRIX, datová matice, která je jednak adekvátní pro oblast vícerozměrné analýzy, jednak se používá pro výstup výsledků,

SERIES, časová řada, což je vhodný datový typ pro práci s posloupností dat.

a) Typ MATRIX

Základní komponenty typu MATRIX jsou

title, titulek, text popisující datovou matici,
nrow, počet řádků,
ncol, počet sloupců,
mode, typ datové matice, může se jednat zejména o typy
    general, obecná čtyřúhelníková matice,
    symmetric, symetrická matice,
    diagonal, diagonální matice, atd.
rowname, dynamické pole obsahující názvy řádků,
colname, dynamické pole obsahující názvy sloupců,
e, jednotlivé položky matice.

Důležité je zejména to, že typ MATRIX je realizován dynamicky, což je sice na jedné straně velmi výhodné z hlediska objemu použitelné paměti, ale na druhé straně je třeba respektovat některé elementární pojmy z oblasti objektového programování (i když to takto ještě programováno není). Nad datovou maticí je deklarována celá řada operací (funkcí a procedur), které lze rozdělit do několika kategorií:

a) *Konstruktory*, pomocí kterých vytváříme nové datové matice.

Patří sem zejména tyto prodprogramy

function *LiteralMatrix*(st: string): matrix; která vytváří datovou matici z textu (prvky řádku oddělené mezerou, řádky oddělené /),

function *ReadMatrix*(soubor: string): matrix; která čte datovou matici z ASCII souboru "inteligentním způsobem"; je však vhodné dodržet posloupnost
titulek (název matice)
typ matice (např. general)
počet řádků a počet sloupců
dále už funkce sama pozná, zda jsou udány názvy řádků a sloupců a přečte jednotlivé prvky,

function *AllocateMatrix*(title: string; nrow,ncol: integer; mode: matrix_mode_type); která vytvoří "kostru" matice daného typu s příslušným názvem a počtem řádků a sloupců;

b) Destruktory, kam lze zařadit zejména proceduru

free_matrix(x: matrix); která uvolňuje dynamickou paměť přiřazenou matici x;

V současném stavu ESTATu je nutno věnovat zvýšenou pozornost tomu, aby byly vždy datové matice řádně vytvořeny a naopak uvolňována jimi obsazovaná paměť!

c) Podprogramy pro čtení a modifikaci prvků matice, kam patří zejména následující

komponenta	funkce pro čtení	procedura pro modifikaci
prvek	get(X,r,c): real	let(X,r,c,Real)
název sloupce řádku titulku	 colname(X,c):string rowname(X,r):string GetTitle(X) :string	 let_colname(X,c,string) let_rowname(X,r,string) let_title(X,string)
počet sloupců řádků	 ncol(X): integer nrow(X): integer	
typ	ModeMatrix: string	

kde X je proměnná typu MATRIX, r je řádek a c je sloupec.

b) Typ SERIES

Datový typ SERIES je realizován klasickým způsobem jako record s komponentami

Title : string , obsahuje název časové řady,
Year : word , počáteční rok,
Period: word , periodicita,
Start_period , počáteční období (má význam pouze u krátkodobých časových řad),
Items , počet pozorování,
item: array [1..200] or real; jednotlivé prvky časové řady.

Základní datové typy a procedury pro práci s časovými řadami jsou obsaženy v programové jednotce *Useries*. Vzhledem ke klasickému způsobu práce jsou jednotlivé komponenty jednoduše přístupné i bez použití podprogramů.

4. Logika konstrukce uživatelského rozhraní

Základem komunikace s uživatelem je obrazovka, takže styl práce se v podstatě určen tím, jaké je její základní rozvržení. V ESTATu pokud možno dodržujeme tyto konvence:

první řádek	nápověda ke způsobu ovládání
druhý řádek	komentář / jednoduché odpovědi
poslední řádek	informace o delších činnostech

Tímto způsobem pracuje většina novějších podprogramů systému. Nejjednodušší z nich jsou určeny pro konverzaci s uživatelem:

Pro práci s druhým řádkem jsou určeny funkce

```
function AnswerInteger(Text:string; Default: integer):integer;
function AnswerReal(Text:string; Default: real): real;
function AnswerText(Text:string; Default: string;
                    Length: integer): string;
```

které zobrazují na druhém řádku Text a defaultovou hodnotu, kterou lze editovat.

Obdobně procedura

```
message(Text: string);
```

zobrazuje na Text na posledním řádku obrazovky.

Systém je konstruován převážně na nabídkovém principu, pro který jsou k dispozici tři základní funkce. Jejich výsledkem je kromě zobrazení a ovládání nabídky číslo zvolené nabídky (resp. 0 při nezvolení žádné). Jsou to

SmallMenu , základní procedura systému pro ovládání vertikálních nabídek, spojená s nápovědou uloženou v externím souboru .HLP.

GetCommentMenu, funkce pro ovládání horizontálních nabídek, kde se ve druhém řádku objevuje komentář,

GetMenu, funkce pro vytváření nabídky z mnoha možností.

Funkce SmallMenu a GetCommentMenu jsou vytvářeny přímo z textu a jsou tedy do značné míry samopopisné, což výrazně přispívá k čitelnosti výsledného programu. Výběr se provádí pomocí kurzorových kličů nebo prvního písmene nabídky.

Každý komplexnější program v ESTATu by měl začínat hlavičkou, která zároveň integruje funkci SmallMenu s nápovědou:

Estat_face(hlavička, program: string);
v rámci této procedury se formuje vstupní obrazovka programu či paketu, informativní text (hlavička) v jeho rámci a název programu, který je důležitý pro interaktivní nápovědu uvedenou v souboru NÁZEV.HLP; nápověda reaguje na texty uvedené v položkách SmallMenu,

SmallMenu(z,y: word; text1: string): word;
je základní funkce pro kontrukci nabídek; jejich konstrukce je velmi jednoduchá, udává se pouze poloha na obrazovce (x,z souřadnice a texty jednotlivých nabídek jsou odděleny lomítkem.

5. Nejdůležitější podprogramy

Většina základních procedur je sdružena v programových jednotkách Umajor, UmtxIO, Ugreat a Useries, které jsou nutné pro konstrukci jednoduchých paketů a programů. Jednotka Umajor je povinná pro všechny ostatní programové jednotky.

Pro konstrukci paketů a programů jsou nejdůležitější podprogramy obsažené v Ugreat. Kromě již jmenovaných jsou to zejména

Input_matrix(var fn: string80; var x: matrix);
Jedná se o základní proceduru pro vstup datové matice, která ovládá jednak výběr ze souborů na disku včetně změny drivu, jednak umožňuje i přechod do vstupu z klávesnice;

Spread_matrix(var y: matrix; var fn: string80;
memo: global_procedure; transform: boolean);

Tato procedura obsahuje specializovaný tabulkový editor a má klíčový význam pro vstup i výstup údajů. Obsahuje vše, co pokládáme za nezbytné pro tyto účely.

Procedura slouží k zobrazení datové matice y po stránkách, a ovládá se způsobem běžným u tabulkových procesorů. Jed-

notlivé prvky datové matice jsou tedy políčka, mezi kterými se lze pohybovat pomocí kurzorových kláves a případně lze upravovat jednotlivé hodnoty.

Kromě editování datové matice jsou k dispozici i specializované příkazy, ovládané pomocí klíčů:

F1 Help - vysvětlení činnosti editoru, které má dvě dílčí nabídky:
Control - vysvětlení ovládání tabulkového editoru,
Memo - procedura pracující s maticí y, uvedená při vyvolání *spread_matrix* a ve které je kontextový komentář k zobrazované matici (předpokládáme, že zde bude zejména elementární vysvětlení výsledků),
F2 Titles - změna názvů řádků, sloupců a tabulky,
F3 Insert - vložení řádku či sloupce,
F4 Delete - vymazání řádku či sloupce,
F5 Format - přeformátování tabulky,
F6 Generate - vyplnění sloupce aritmetickou posloupností,
F7 re-Compute - přepočtení sloupce, v jejímž rámci lze použít Pascalovských matematických funkcí, závorek atd.,
F8 Procedures - operace s datovou maticí, což jsou
Zscore - normování sloupců,
Cols - výběr určitých sloupců,
Rows - výběr určitých řádků,
Miss - vyloučení řádků s chybějícími pozorováními,
Trans - transpozice, záměna řádků a sloupců,
F9 Print - tisk datové matice ve zvolené šířce tiskárny,
F10 Save - ukládání datové matice do souboru fn nebo do nového souboru.

Pokud je hodnota booleovské proměnné Transform pravdivá, jsou k dispozici všechny tyto možnosti. V opačném případě jsou zakázány některé změny (tedy editování a klíče 3,4,6,7,8), aby nedocházelo např. ke zkreslování výsledků.

spread_series(var y: series; fn: string80; bol: boolean);
Procedura slouží k prezentaci a úpravám časové řady, je určitým ekvivalentem předchozí procedury a je založena na podobných zásadách.
Význam ovládacích klíčů je následující:

F1 Help - vysvětlení práce s editorem,
F2 Name - změna názvu řady,
F3 Insert - vložení údaje na dané místo (s posunem ostatních),
F4 Add - přidání údajů na konec řady (s posunem),
F5 Del - vypuštění údaje,
F6 reCompute - přepočtení údajů,
F7 Format - změna šířky všech sloupců,
F8 Graph - grafické znazornění řady,
F9 Print - tisk řady ve formě matice,
F10 Save - uložení ve zvolené formě.

S časovými řadami pracujeme paralelně ve dvou formách
- jsou uloženy v proměnné typu *SERIES* (*.TIM),
- zobrazujeme je ve tvaru dvourozměrné datové matice a pro tyto účely je vnitřně ukládáme do proměnné typu *MATRIX* (*.MAT).

6. Příklad aplikace

Pro flexibilní konstrukci jednoduchých paketů je vhodné, když jsou jednotlivé komponenty konstruovány ve formě procedur nad datovou maticí. Následující příklad realizace velmi jednoduchého a přitom již poměrně silného paketu pro vícerozměrnou analýzu ilustruje tuto filozofii:

```
program Multi;
{$I ESTAT.INC }
uses Crt, Dos, Printer, Umajor, UmtxIO, Umatrx, Umulti, Ugreat;
var     i,j : integer;
         s  : string;
         ch : char;
  InputFile, OutputFile : string80;
        x,y : matrix;

label  Start, Data, Actions;

{begin of main program}
begin
 Start: Estat_face('Vicerozmerna analyza','Multi');
        InputFile:=''; OutputFile:=''; x:=nil; y:=nil;

 Data: ClrScr;
  case SmallMenu(5,20,
  'Ilustracni priklad/Vstup z klavesnice/Vstup ze souboru')
  of
  0: goto Start;
  1: begin free_matrix(y); InputFile:='LIT.MAT';
     y:=literal_matrix('1 2 3 4 5/10 20 30 40');end;
  2: begin free_matrix(y); InputFile:='NEW.MAT';
     Input_matrix(InputFile,y); end;
  3: Input_matrix(InputFile,y);
  end;

 Actions: ClrScr;
  gotoXY(15,2); write('Volba statisticke metody');
 case SmallMenu(5,20,
 'Uprava dat/Zakladni statistiky/Regresni analyza/Shlukova ana-
  lyza/Metody razeni/Analyza hlavnich komponent/Nova data')
  of
 1: spread_matrix(y,OutputFile,default_memo,True);
 2: begin x:=StatsMatrix(y);
    spread_matrix(x,OutputFile,default_memo,false);
    free_matrix(x); end;
 3: Choice_regression(y);
 4: begin Hierarchical_clustering(y,InputFile,1);
    goto Data; end;
 5: begin Ordering_methods(y,InputFile); goto Data; end;
 6: begin ClrScr; Writeln('Pocitam hlavni komponenty');
    x:=Principal_components(y);
    spread_matrix(x,OutputFile,default_memo,False);
    free_matrix(x);  end;
0,7: begin InputFile:=''; free_matrix(y); goto Data; end;
 end;
 goto Actions;
 free_matrix(y);
end.
```

7. Podprogramy pro konstrukci algoritmů

Pro podporu konstrukce vlastních algoritmů obsahuje ESTAT velmi mnoho prostředků jak z oblasti numerických metod, tak z oblasti statistikých algoritmů. Na tomto místě se zastavíme pouze u některých aspektů této problematiky.

a) Chybějící pozorování

Většina komplexnějších podprogramů pracuje správně i s chybějícími pozorováními (nebo jsou postupně upravovány).

K dispozici jsou zejména

global_missing_text, text zobrazovaný v případě chybějícího pozorování,

function *missing*: real , jejímž výsledkem je systémové zobrazení chybějícího údaje,

function *IsMissing*(r: real): boolean , test, zda je je určité číslo chybějící,

function *Divide*(a,b: real): real , jejímž výsledkem je chybějící hodnota když a nebo b jsou chybějící, nebo b je rovno 0.

b) Interakce s klávesnicí

Znak z klávesnice se čte prostřednictvím funkce

```
function Scanner : char;
```

která umožňuje rozlišit i kombinace kličů.

Jednotlivé kombinace lze pak identifikovat pomocí názvů příslušných konstant:

```
ESC     = #27;  NULL     = #0;   ENTER=#13;
FNK1    = #187; FNK2     = #188; BkSp =#8;
FNK3    = #189; FNK4     = #190; CtrlHome = #247;
FNK5    = #191; FNK6     = #192; CtrlEnd  = #245;
FNK7    = #193; FNK8     = #194; CtrlPgUp = #4;
FNK9    = #195; FNK10    = #196; CtrlPgDn = #246;
InsKey  = #210; DelKey   = #211; CtrlLeft = #243;
UpKey   = #200; DownKey  = #208; CtrlRight= #244;
LeftKey = #203; RightKey = #205;
PgUpKey = #201; PgDnKey  = #209;
HomeKey = #199; EndKey   = #207;
Tab     = #9;   BackTab  = #143;
```

c) Tisk

Vzhledem k tomu, že zatím Slušovice nedodaly pro počítače VŠE tiskárny, pracujeme ve velmi obtížných podmínkách a jenom příležitostně používáme tiskárnu EPSON. Pro ovládání tiskárny jsou k dispozici konstanty:

```
global_switch_to_graphic : string = ESC+'t1';
global_switch_back       : string = ESC+'t0';
New_page                 : string = chr(12);
gpage_width      : integer  = 80;
gpage_size       : integer  = 63;
```

d) Barvy

V současné době máme určité zkušenosti pouze s grafickými kartami CGA, EGA a Hercules, z nichž barevná je pouze EGA. Přesto rozlišujeme následující typy barev (s uvedením nastavení pro EGA):

Základní barvy:
```
gcolor_back    := blue;
gcolor_message:= green;
gcolor_data    := yellow;
gcolor_input   := white;
gcolor_error   := red;
```

Inverzní barvy:
```
gincolor_back    := lightgray;
gincolor_message := blue;
gincolor_data    := black;
```

Barvy pro grafiku:
```
gdrcolor_back    := white;
gdrcolor_drawing:= blue;
gdrcolor_message:= red;
gdrcolor_axis    := black;
```

Pro zjednodušení práce s barvami je vhodné používat procedur *NormColor;* která nastavuje základní barvy (back a data), *InvColor;* která nastavuje inverzní barvy..

Nastavení je automatické v programové jednotce Umajor. V této souvislosti je nutno upozornit i na počáteční adresu grafické karty uvedenou v konstantě

```
global_monitor_offset  : word     = $b800; ( pro CGA )
                                    $a000; ( pro EGA )
                                    $b000; ( pro Hercules )
```

8. Vytváření programů

V současné době ještě není vybavení statistiků počítači ani zdaleka srovnatelné se světem, takže snad každý narazí při vytváření programů v ESTATu na některé kapacitní problémy. Z těchto důvodů se musíme zastavit i u této problematiky.

Prakticky nelze pracovat při současném použití rezidentních programů (Norton, XTPRO atd.). Ve většině případů se nám dokonce nepodaří přeložit program bez zásahu do nastavení kompilátoru. Jediným naprostým nutným nastavením je

Force Far Calls On , pro spolupráci vzdálených podprogramů,

Overlays Allowed On , protože bude využito perspektivně překrývání programových jednotek.

Ostatní nastavení je možno vypnout (*Off*), čímž se velmi podstatně ušetří vnitřní i vnější paměť.

Ve většině případů je také nutné provádět kompilaci na disk. Výsledné programy nejsou zpravidla příliš rozsáhlé, protože kompilátor je velmi efektivní a nevyužívaný kód se v něm neobjevuje.

Někdy je dokonce nutné využít místo programovacího prostředí *TURBO.EXE* řádkový kompilátor *TPC.EXE*, který vyžaduje podstatně méně vnitřní paměti.

9. Závěr

Systém ESTAT má ještě velmi daleko do dokonalosti a jistě nelze ani tvrdit, že má profesionální programátorskou úroveň. To však nebylo zatím cílem, není určen pro profesionály v programování, ale pro statistiky, kteří by rádi předložili své metody veřejnosti.

Domníváme se, že má budoucnost, pokud se prací zúčastní více lidí, a zejména pokud budou rozumnou formou spolupracovat.