V. évfolyam, 6. szám
Ára: 299 Ft

Számítástechnika haladóknak

Computer PANORÁMA

94. június

Teszt: grafikus kártyák

Isteni színjáték

CD-ROM melléklet

Új hajtások

486-os processzorok

Eladósorban

Öko PC-k

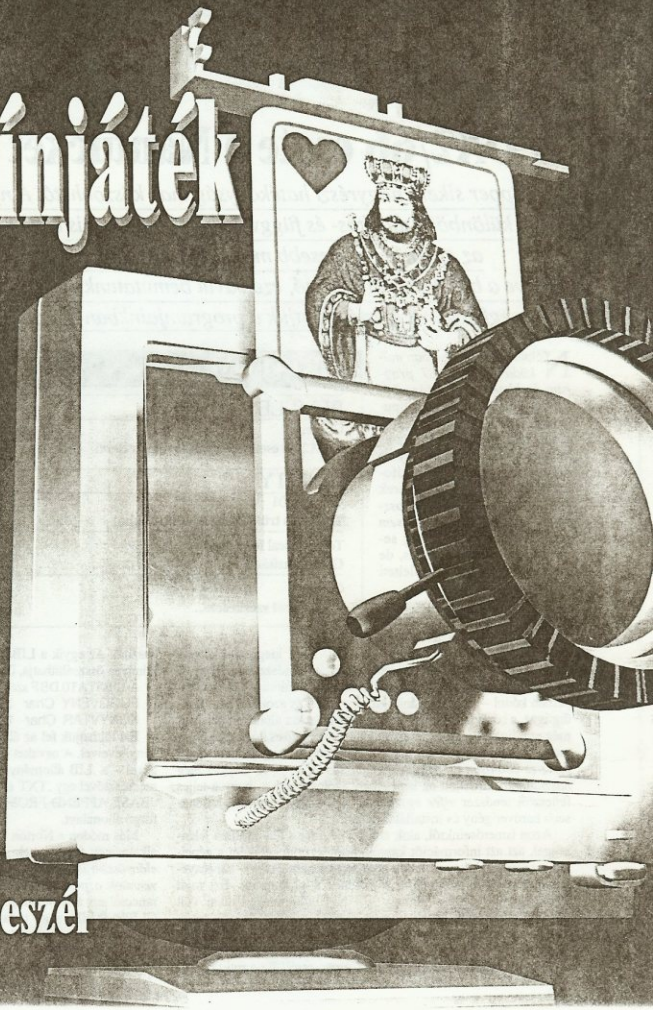
Zöld út

Texas notebook

Utazó ügynök

HP LaserJet 4M Plus

Több nyelven beszél



SZOFTVER ÚJSÁG
Computer
PANORÁMA

Clipper

A DX2/66 esete a Nantucket Tools II-vel

A Clipper sikere nagyrészt hatékonyságának köszönhető. A nyelvet a megjelenése óta különböző utasítás- és függvénykönyvtárakkal is kiegészítették. Írásunkban az egyik legsikeresebb modul és a gyors processzorok kapcsolatáról, illetve a bővítésekről lesz szó, ezenkívül bemutatunk egy olyan alkalmazást, amelynek segítségével kiértékelhetjük a programjainkban használt külső könyvtárakat.

Néhány, *eddig kiválóan működő Clipper '87 program újabban komoly aggodalmakra ad okot.* A mai géptípusokon, a DX2-es 486-os processzorok 50 és 66 MHz-es változataival felszerelt gyors számítógépeken az eddig kifogástalanul működő rendszerek *menet közben lefagynak, roszszabb esetben pedig el sem indulnak.* Helyenként még segít a turbógombos lassítás, de ez is csak a lefagyást késlelteti néhány perccel.

Az ok egyelőre ismeretlen, és mindössze annyit tudunk, hogy ha az alapkészleten és a saját függvényeinken (Clipper, C, ASM) kívül kiegészítő könyvtárakat is használunk, akkor a rendelleneség rendszerint fellép.

Elég „elfajzott" programozó az, aki még mindig '87-es Clipperben kódol – mondhatnánk erre. Ez az állítás azonban nem mindig igaz, s bár a cikk szerzője a verzióváltásokat általában helyesli, még mindig sok esetben találkozunk Clipper '87-es kóddal. E programozók mentségére legfeljebb az hozható fel, hogy például *egy 8 MHz-es 286-oson 80 Mbájtos winchesterrel a '87-es Clipper még használható.* Emellett az sem elhanyagolható előny, hogy a teljes fejlesztői rendszer *elfér egyetlen HD-s lemezen,* minden különösebb hardverigény és installálás nélkül.

Azon ismerőseinktől, akik találkoztak már a fent említett jelenséggel, azt azt információt kaptuk, hogy gyorsségként a gépeiken a setupban kikapcsolták a memória cache-t, és így – bár lényegesen lassabban – egyelőre működnek a programok. Ezt saját tapasztalataink alapján mintegy 90%-ban tudjuk megerősíteni: volt ugyanis olyan gép, amelyen nem segített a trükk. Kérjük tehát, hogy jelentkezzen, aki a szóban forgó gondról többet tud, esetleg megfejtette már a hiba okát!

És most néhány szó a LIBSTAT programról! Ez a rutin bármilyen Clipper '87-es forrásprogramot végignéz, és megmondja, milyen könyvtárakra építkezve készítették azt. *A felismerhető könyvtárak: CLIPPER, EXTEND, MGV 87 STOOLS, NANTUCKET TOOLS II, NETLIB, PROCLIB 87, RAMBO LIB, TOM RETTIG LIB.* A felsoroltakon kívül természetesen más könyvtárakat is azonosíthatunk, de a programot ehhez bővíteni kell.

Általában arra az információra van szükségünk, hogy hány és milyen függvényt hívunk az alapkészleten kívül.

A program két adatállományt használ. Az egyik a LIBSTAT0.DBF, amelynek a tartalmát bárki könnyen összeállíthatja, illetve módosíthatja.

A LIBSTAT0.DBF szerkezete:

```
FUGGVENY Char  30  0
KONYVTAR Char  12  0
```

Ezt töltsük fel az általunk használt Clipper könyvtárak függvényneveivel. A neveket – a LIB.EXE könyvtármenedzser segítségével – a .LIB állományokból vehetjük ki, és bármilyen szövegszerkesztővel egy .TXT állományba szerkeszthetjük azokat, ahol a dBASE APPEND FROM ... SDF utasítással bővíthetjük a szóban forgó állományt.

Más módon a Norton Guide-ok visszafordításával juthatunk az alkalmazott függvények neveihez. *Ez utóbbi megoldás talán még előnyösebb is, hiszen az előbb a CLIPPER.EXE fájlból ki kellett vennünk a parancsneveket* (a Norton Commander F3-View paranccsal ezt a területet az állomány vége felé találjuk), ugyanis a CLIPPER.LIB nem tartalmazza mindegyiket. *Figyeljünk arra, hogy a Clipper alaputasításokat négy betűvel rövidíteni lehet,* tehát ezeket is bele kell vennünk a készletbe.

Ezenkívül már csak egy adatfájlra van szükség, nevezzük ezt

LIBWORK.DBF-nek! Ennek a szerkezete a következő:

SOR	Char	254	0
UTASITAS	Char	100	0

E szerkezet másolataként – mintegy mellékterméként – létrejön a vizsgálandó programfájl nevét viselő, ám .DBF kiterjesztésű állomány, amelyben még azt is ellenőrizhetjük, hogy melyik függvénynév alapján azonosítottunk egy könyvtárat.

A program az indulásakor bekéri a vizsgálandó forrás nevét (legfeljebb 8 karakter hosszú kiterjesztés nélkül), majd automatikusan indexel, ha nem találja meg az indexállományt. Két menetben elemzi a sorokat, és a már említett fájlnév.DBF-en kívül két szövegfájlt kapunk eredményül: a fájlnév.LBS-et és a fájlnév.LBX-et. A futás végén egy statisztika is megjelenik az analizált kódban használt függvények hovatartozásáról.

Az .LBS a LIBSTAT-tal generált, a forrásállománnyal megegyező, de kommentezett szöveg, amely a következő szabályok szerint keletkezik:

– Ha a keresett függvénynév kommentben szerepel, akkor a program ezt az előfordulást nem veszi figyelembe (a komment jelölése lehet sorvégi '&&' vagy sor eleji '*' jel); például a * CLEA SCRE-nek a kommentezés miatt nincs hatása.

– Ha a megtalált függvénynév sora 50 karaktemél rövidebb, akkor a program ezt egy kommentben – '&& __ KÖNYVTÁRNÉV' – jelzi az 52. karakterhelyen. Például:

set curs off && kurzor ki && __ EXTEND,CLIPPER

– Ha a megtalált függvénynév sora 50 karaktemél hosszabb, akkor a program ezt egy kommentben – '&& __ KÖNYVTÁRNÉV' – jelöli a sor végén.

– Ha egy sorban többféle könyvtár függvényei is előfordulnak, akkor a program egy kommentben jelöli ezeket, és vesszővel választja el a könyvtárneveket. Például:

fwrite(handl2, string + sorveg, len(string) + 2) && __ CLIPPER,CLIPPER

Az .LBX a LIBSTAT-tal generált, és csak az alapkészleten kívüli hívásokat tartalmazó szöveg, amely a következő szabályok szerint keletkezik:

– Ha nem volt alapkészleten kívüli hívás, akkor a szöveg csak a statisztikát tartalmazza. Például:

Statisztika:

Clipper LIB.....: 137 db
Extend LIB.....: 15 db
MGV 87 (Stools).: 0 db
Nantucket Tools II : 0 db
NetLIB.....: 0 db
Proclib 87.....: 0 db
Rambo LIB.....: 0 db
Tom Rettig LIB.....: 0 db

– Ha alapkészleten kívüli hívás is volt, akkor a program az eredeti forrásprogram szerinti sorszáma után írja ki a felismerés alapjául szolgáló sor tartalmát. Például:

00077: tokeninit(@qsor, elvalaszt) && __ NT2

Ha az idegen könyvtárak funkcióit a jövőben saját függvényekkel akarjuk kiváltani, akkor ez a módszer nagy segítséget adhat.

Csizmazia István

A HBSTAT.PRG forráslistája

```
*-----------------------------------------*
*   Főprogram neve.....: LIBSTAT
*   Szerző.....: Csizmazia D. István (C)
RamboSoft Works
*   Megrendelő.....:
*   A főprogram feladata....: milyen Clipper
könyvtári függvényeket használtak
*
*
*   Elkezdve.....: 1994.03.25.
*   Paramétert kap.....:
*   Paramétert ad.....:
*   Hívója.....: A DOS
*   Használt naplójelei.....:
*   Az itt megírt függvények:
*   Megjegyzés.....:
*   Kell majd.....:
*
*
*


set softseek off
set exact on
sorveg = chr( 13 ) + chr( 10 )  && EOL jel

*         be- és kimenő file nevek bekérése
*
clea scre
xinput = spac( 8 )
@ 01, 00 say 'Kérem a CLIPPER forrás file nevét ( pl.
PROGRAM )' get xinput
read
set curs off

*         használt állománynevek generálása
*
prgnev = xinput + '.prg'
outnev = xinput + '.lbs'
outne2 = xinput + '.lbx'
dbfnev = xinput + '.dbf'

*         ha nem létezik, akkor a viszonlátásra
*
if ! file( &prgnev' )
   @ 03, 00 say 'Nem létező file...'
   ?
   quit
endi
```

```
sele 1
use libwork
copy to &dbfnev
use &dbfnev alias dbfnev
zap

appe from &prgnev sdf
wrecc = recc()

sele 2
use libstat0 alias libstat0
lrecc = recc()
if ! file( 'libstat0.ntx' )
   @ 03,00 say 'Indexelés folyik...'
   index on fuggveny to libstat0
endi
set index to libstat0

*         a tokeninit szóelválasztásához
*
elvalaszt = chr(  0 ) + chr(   9 ) + chr(  10 ) +
chr( 13 ) + chr( 26 ) ;
          + chr( 32 ) + chr( 138 ) + chr( 141 ) ;
          + ',.;:|()~'"#^'

sele dbfnev
go top
for i = 1 to wrecc
    qsor = uppe( rtrim( sor ) )

    poz = at( '&&', qsor )                    && komment-
ben nem nézünk
    if poz > 0
       qsor = left( qsor, poz - 1 )
    endi

    if ( left( ltrim( qsor), 1 ) != '*' ) && komment-
ben nem nézünk
       j = numtoken(sor)
       tokeninit( @qsor, elvalaszt )

       @ 05,00 say 'Sorok analizálása.: ' + sform(
wrecc ) + ' / ' + sform( i )
       store '' to whonnan
       store '' to xfuggveny
       sele libstat0
       go top

       if j > 0
          wkeres   = tr_wpstrip( uppe( tokennext() )
)
```

```
        do while ! empty( wkeres )
           seek wkeres
           if found()
              wfuggveny = fuggveny
              wfuggveny = alltrim( fuggveny )

              if ( wkeres == wfuggveny )
                 wkonyvtar = alltrim( konyvtar )
                 xfuggveny = xfuggveny + wfuggveny
+ ','
                 store alltrim( whonnan ) + wkonyv-
tar + ',' to whonnan
                 endi
              endi

              wkeres   = tr_wpstrip( uppe( tokennext()
) )
           endd
        endi

        sele dbfnev
        if ! empty( whonnan )
           if right( whonnan, 1 ) = ','
              whonnan = left( whonnan, len( whonnan )
- 1 )
           endi

           if right( xfuggveny, 1 ) = ','
              xfuggveny = left( xfuggveny, len( xfug-
gveny ) - 1 )
           endi

           repl utasitas with xfuggveny
           whonnan = ' && __ ' + alltrim( whonnan )
           xsor = sor
           xsor = rtrim( xsor )
           if len( xsor ) < 50
              xsor = pad( xsor, 50 )
           endi

           repl sor with xsor + whonnan
        endi
     endi
     skip
next

sele dbfnev
go top

*          prg file commentekkel módositva
*
handle = fcreate( outnev, 0 )
handle = fopen( outnev, 2 )

*           különleges sorok külön file-ba irása és
statisztika          *
handl2 = fcreate( outne2, 0 )
handl2 = fopen( outne2, 2 )

egyeb = .f.
sum_cl = 0
sum_ex = 0
sum_nt = 0
sum_tr = 0
sum_ra = 0
sum_mg = 0
sum_pr = 0
sum_ne = 0

for i = 1 to wrecc
    @ 07,00 say 'Textfile generálás: ' + sform( wrecc
) + ' / ' + sform( i )
    qsor = sor
    qsor = rtrim( qsor ) + sorveg
    fwrite( handle, qsor, len( qsor ) )
    if at( '&& __', qsor ) > 0
       do case
          case ( at( '&& __ NT2'    , qsor ) > 0 )
               sum_nt = sum_nt + 1
               egyeb = .t.

          case ( at( '&& __ MGV87'  , qsor ) > 0 )
               sum_mg = sum_mg + 1
               egyeb = .t.

          case ( at( '&& __ PROCLIP', qsor ) > 0 )
               sum_pr = sum_pr + 1
               egyeb = .t.

          case ( at( '&& __ NETLIB' , qsor ) > 0 )
               sum_ne = sum_ne + 1
               egyeb = .t.

          case ( at( '&& __ RAMBO'  , qsor ) > 0 )
               sum_ra = sum_ra + 1
               egyeb = .t.

          case ( at( '&& __ TOM R.' , qsor ) > 0 )
               sum_tr = sum_tr + 1
               egyeb = .t.

          case ( at( '&& __ CLIPPER' , qsor ) > 0 )
               sum_cl = sum_cl + 1

          case ( at( '&& __ EXTEND' , qsor ) > 0 )
               sum_ex = sum_ex + 1
       endc

       if egyeb
          egyeb = .f.
          sorsz = strzero( i, 5 ) + ': '
          qsor = sorsz + qsor
          fwrite( handl2, qsor, len( qsor ) ) && újra
a másikba is
       endi
    endi
    skip
next

fclose( handle )

close all

string = sorveg                          && soremelés
fwrite( handl2, string, len( string ) )

string = '          Statisztika:'
@ 09, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = '-----------------'
@ 10, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Clipper LIB........: ' + pad( sform( sum_cl
), 4 ) + ' db'
@ 11, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Extend LIB.........: ' + pad( sform( sum_ex
), 4 ) + ' db'
@ 12, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'MGV 87 ( Stools )..: ' + pad( sform( sum_mg
), 4 ) + ' db'
@ 13, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Nantucket Tools II.: ' + pad( sform( sum_nt
), 4 ) + ' db'
@ 14, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'NetLIB.............: ' + pad( sform( sum_ne
), 4 ) + ' db'
@ 15, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Proclip 87.........: ' + pad( sform( sum_pr
), 4 ) + ' db'
@ 16, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Rambo LIB..........: ' + pad( sform( sum_ra
), 4 ) + ' db'
@ 17, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

string = 'Tom Rettig LIB.....: ' + pad( sform( sum_tr
), 4 ) + ' db'
@ 18, 00 say string
fwrite( handl2, string + sorveg, len( string ) + 2 )

*          minden becsuk, kurzor vissza, elköszön
*
fclose( handl2 )
?
?'Csao ragazzi!'
set curs on
retu
```

LIBWORK.DBF-nek! Ennek a szerkezete a következő:

SOR	Char	254	0
UTASITAS	Char	100	0

E szerkezet másolataként – mintegy mellékterméként – létrejön a vizsgálandó programfájl nevét viselő, ám .DBF kiterjesztésű állomány, amelyben még azt is ellenőrizhetjük, hogy melyik függvénynév alapján azonosítottunk egy könyvtárat.

A program az indulásakor bekéri a vizsgálandó forrás nevét (legfeljebb 8 karakter hosszú kiterjesztés nélkül), majd automatikusan indexel, ha nem találja meg az indexállományt. Két menetben elemzi a sorokat, és a már említett fájlnév.DBF-en kívül két szövegfájlt kapunk eredményül: a fájlnév.LBS-et és a fájlnév.LBX-et. A futás végén egy statisztika is megjelenik az analizált kódban használt függvények hovatartozásáról.

Az .LBS a LIBSTAT-tal generált, a forrásállománnyal megegyező, de kommentezett szöveg, amely a következő szabályok szerint keletkezik:

– Ha a keresett függvénynév kommentben szerepel, akkor a program ezt az előfordulást nem veszi figyelembe (a komment jelölése lehet sorvégi '&&' vagy sor eleji '*' jel); például a * CLEA SCRE-nek a kommentezés miatt nincs hatása.

– Ha a megtalált függvénynév sora 50 karaktemél rövidebb, akkor a program ezt egy kommentben – '&& __ KÖNYVTÁRNÉV' – jelzi az 52. karakterhelyen. Például:

set curs off && kurzor ki && __ EXTEND,CLIPPER

– Ha a megtalált függvénynév sora 50 karaktemél hosszabb, akkor a program ezt egy kommentben – '&& __ KÖNYVTÁRNÉV' – jelöli a sor végén.

– Ha egy sorban többféle könyvtár függvényei is előfordulnak, akkor a program egy kommentben jelöli ezeket, és vesszővel választja el a könyvtárneveket. Például:

fwrite(handl2, string + sorveg, len(string) + 2) && __ CLIPPER,CLIPPER

Az .LBX a LIBSTAT-tal generált, és csak az alapkészleten kívüli hívásokat tartalmazó szöveg, amely a következő szabályok szerint keletkezik:

– Ha nem volt alapkészleten kívüli hívás, akkor a szöveg csak a statisztikát tartalmazza. Például:

Statisztika:

Clipper LIB.....: 137 db
Extend LIB.....: 15 db
MGV 87 (Stools).: 0 db
Nantucket Tools II : 0 db
NetLIB.....: 0 db
Proclib 87.....: 0 db
Rambo LIB.....: 0 db
Tom Rettig LIB.....: 0 db

– Ha alapkészleten kívüli hívás is volt, akkor a program az eredeti forrásprogram szerinti sorszáma után írja ki a felismerés alapjául szolgáló sor tartalmát. Például:

00077: tokeninit(@qsor, elvalaszt) && __ NT2

Ha az idegen könyvtárak funkcióit a jövőben saját függvényekkel akarjuk kiváltani, akkor ez a módszer nagy segítséget adhat.

Csizmazia István

A HBSTAT.PRG forráslistája

```
*-----------------------------------------*
*   Főprogram neve.....: LIBSTAT
*   Szerző.....: Csizmazia D. István (C)
RamboSoft Works
*   Megrendelő.....:
*   A főprogram feladata....: milyen Clipper
könyvtári függvényeket használtak
*
*
*   Elkezdve.....: 1994.03.25.
*   Paramétert kap.....:
*   Paramétert ad.....:
*   Hívója.....: A DOS
*   Használt naplójelei.....:
*   Az itt megírt függvények:
*   Megjegyzés.....:
*   Kell majd.....:
*
*
*

set softseek off
set exact on
sorveg = chr( 13 ) + chr( 10 )  && EOL jel

*         be- és kimenő file nevek bekérése
*
clea scre
xinput = spac( 8 )
@ 01, 00 say 'Kérem a CLIPPER forrás file nevét ( pl.
PROGRAM )' get xinput
read
set curs off

*         használt állománynevek generálása
*
prgnev = xinput + '.prg'
outnev = xinput + '.lbs'
outne2 = xinput + '.lbx'
dbfnev = xinput + '.dbf'

*         ha nem létezik, akkor a viszonlátásra
*
if ! file( &prgnev' )
   @ 03, 00 say 'Nem létező file...'
   ?
   quit
endi
```

```
sele 1
use libwork
copy to &dbfnev
use &dbfnev alias dbfnev
zap

appe from &prgnev sdf
wrecc = recc()

sele 2
use libstat0 alias libstat0
lrecc = recc()
if ! file( 'libstat0.ntx' )
   @ 03,00 say 'Indexelés folyik...'
   index on fuggveny to libstat0
endi
set index to libstat0

*         a tokeninit szóelválasztásához
*
elvalaszt = chr(  0 ) + chr(   9 ) + chr(  10 ) +
chr( 13 ) + chr( 26 ) ;
          + chr( 32 ) + chr( 138 ) + chr( 141 ) ;
          + ',.;:|()-~"#^'

sele dbfnev
go top
for i = 1 to wrecc
    qsor = uppe( rtrim( sor ) )

    poz = at( '&&', qsor )                    && komment-
ben nem nézünk
    if poz > 0
       qsor = left( qsor, poz - 1 )
    endi

    if ( left( ltrim( qsor), 1 ) != '*' ) && komment-
ben nem nézünk
       j = numtoken(sor)
       tokeninit( @qsor, elvalaszt )

       @ 05,00 say 'Sorok analizálása.: ' + sform(
wrecc ) + ' / ' + sform( i )
       store '' to whonnan
       store '' to xfuggveny
       sele libstat0
       go top

       if j > 0
          wkeres   = tr_wpstrip( uppe( tokennext() )
)
```