PROGRÁMOZÁS

FORRÁSKÓD

A programozók lapja

INTERKOMBO KFT
1126 Budapest, Fodor u. 12., mfsz. 4. • Telefon/Fax: 155-1897

ibico® Termékek magyarországi forgalmazója

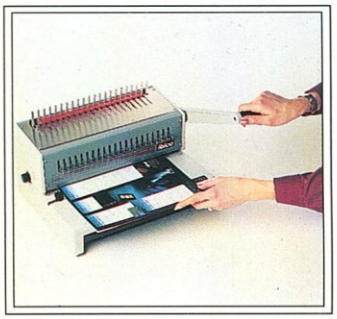
Spirál iratfűzés három lépésben

KOMBO
38.000 Ft

modern
egyszerű
olcsó

Kombo, a svájci csodagép – az üzleti siker titka

Reklám, piackutatás, árlisták, ajánlatok, kimutatások, tervek – könyv formában – az Ön irodájában.



5 – 410 lap

Ø 6 mm – 51 mm

ibico® Hőkötő – a jó üzletkötő

310-E
Hűtőállvánnyal
16.000 Ft

① **Iratokat borítókba**
② **Ezeket kötővályúba helyezzük**
③ **30 másodperc – hangjel – kiemeljük**
④ **Lehűtjük**

Maximum lapszám: 200 (80 g/m³)

Feszültség: 220/240 V 30 W

Súly: 2 kg

Tartalom

Előfizetés

Lapunk tervezett ára	298 Ft.-
Ha Ön előfizeti, kedvezményt	kap.
Előfizetési díj fél évre:	1494 Ft.-
Így Önnek egy szám csupán	249 Ft.-
Előfizetési díj egy évre:	2388Ft.-
Így Önnek egy szám csupán	199 Ft.-
Iskoláknak az előfizetési díj	egy évre: 1188 Ft.-
Így egy szám csupán	99 Ft.-

Előfizethető a mellékelt csekken, vagy átutalási postautalványon a TEX Studió Kft. 218-98172/541-003028-3 számú számlaszámán.

Társkereső

Ön az első magyarnyelvű, programozóknak szóló magazin próbaszámát tartja a kezében. A Forráskód 1995 januárjától havonta 64 oldalon lemezmelléklettel megjelenő szines lap.

A lapindítás célja: fórumot teremteni mindazoknak akiket nem elégítenek ki a vásárolt programkonzervek hanem saját kezükbe veszik a számítógép irányítását. Segítjük a programnyelvek oktatását, támogatjuk a hazai szoftwer fejlesztőket, számukra fórumot, kedvezményes hirdetési lehetőséget biztosítunk. Szervezője, hírdetőtáblája kívánunk lenni a hazai számítógépes amatőr mozgalomnak. Ismertetjük, vizsgáljuk a számítógép használatának lehetőségeit, határait, kapcsolódását a való világgal. Ehhez társakat keresünk!

Keressük:
- az iskola rovatok szerzőit,
- a profikat akiknek már valami sikerült, és hajlandók a tudásukat megosztani velünk,
- a fejlesztő eszközök forgalmazóit, ismertessék portékáikat,
- és mindenkit akinek van egy jó ötlete!

Aki magára ismert hívjon vagy keressen:

Nagy Sándor
Tel: 134-12-73
1085 Budapest
Népszínház u. 31.I.em 4/a

Digitális I/O kártyák programozása

A mai Pc-n zömmel irodai, ügyviteli alkalmazások, játékok futnak, de néhány ma még nem túl elterjedt hardware elemmel kiegészítve a Pc másra is képes. Lapunk ezen hardware elemek programozásáról indít sorozatot. Első két részben a digitális I/O kártyák programozását ismertetjük.

A digitális I/O kártyákat elektromos berendezések be-, és kikapcsolására, be-, vagy kikapcsolt állapot ellenőrzésére, külső eszközök vezérlésére használhatjuk.
Megjelenésüket nézve normál Pc-s bővítő kártyák többnyire 8 bites ISA busszal. A be-, ill. kimeneti jel többnyire TTL szintű, ha nem az, akkor védeni kell a számítógépet az esetleges túláramoktól. Néha a külvilágot kell védeni a számítógéptől, például robbanásveszélyes környezetben. A védelmet opto-csatolós, relés leválasztással oldják meg, terjednek a szilárdtest-relés megoldások. Kis áramok és kevés csatorna esetén a leválasztást ráépítik a kártyára, más esetekben a gépen kívül helyezik el és szalagkábellel csatlakoztatják a kártyához. Egy kártyán 8-tól 144-ig I/O vonal található /mindig nyolc egészszámú többszöröse/, sokszor egyéb áramkörök /pl. számláló időzítő, analóg érzékelők/ társaságában. Működés és programozás szempontjából kétféle kártyát ismerünk: amelyik megszakítást generál és amelyik nem. A megszakítást nem generáló kártyák programozása nem más, mint egy kártyafüggő portcím írása, olvasása, IN és OUT utasítással. A portcímet a kártyán kapcsolókkal vagy jumperekkel választhatjuk ki.

Szokásos címtartományok /hexa/:

200 - 277
300 - 377
380 - 3AF
3C0 - 3CF
3E0 - 3EF

A portra byte-okat írunk ill. onnan byte-okat olvasunk, így egyszerre nyolc csatornát írunk vagy olvasunk. A byte-okat nekünk kell bitekből összeállítani vagy olvasáskor a byte-ot bitekre bontani. Egyes kártyáknál írás vagy olvasás előtt be kell állítani a szükséges üzemmódot, másoknál nem. Az üzemód-beállítás egy un. inicializáló portra küldött inicializáló paraméter írásából áll. Erre vonatkozóan a kártyához kapott leírásból tájékozódhatunk. A megszakítást generáló kártyák lehetőséget biztosítanak eseményvezérelt programok írására, háttérben történő monitorozásra anélkül, hogy a csatornákat folyamatosan le kellene kérdezni. Az egyes csatornákhoz /vagy csoportokhoz/ prioritási szinteket rendelhetünk, letilthatjuk őket. Több megszakítást kérő csatorna esetén megszakítás-maszkot haszálhatunk. Ez azt jelenti, hogy pl. nyolc csatorna esetén a maszk byte adott bitjéhez rendelt csatornáról engedélyezett a megszakítás ha a bit nulla, egyébként tiltott.
A generált megszakítás általában IRQ 2 - 7.
Ebben a cikkben a hagyományos digitális inputkártyával történő háttér-monitorozásra mutatunk példát.
A megszakítást generáló kártyák programozására egy konkrét alkalmazási példán keresztül még visszatérünk.
Az ismertetett C nyelvű program használható szerviz célra vagy egyszerűbb feladatok figyelésére.
A program rezidensen bent ül a memóriában és másodpercenként 18-szor megkapja a vezérlést az 1c /hexa/ interrupton keresztül.
Ha kell, inicializáló jelet küld a vezérlő portra, majd olvassa az adott címen levő portokat. A kapott értékeket bitekre bontja majd és egy stringbe írja, végül kiírja a stringet a színes karakteres képernyőre. A program nem vizsgálja saját példányainak létét a memóriában, onnan eltávolítani csak külső programmal, vagy a gép újra indításával lehet. Kiíratásnál nem figyeljük az elektronsugár visszafutását.
A közölt listát bárki szabadon módosíthatja, az említett szolgáltatásokkal kiegészítheti, de számolnia kell az erőforrás-igények ugrásszerű növekedésével. A programot lehetőleg COM formátumra fordítsuk. Turbo C 2.0 -val fordítva 8560 byte méretű programot kapunk, ami futáskor kb. 10 Kbyte helyet foglal a memóriából.
Egy példa az alkalmazásra:
Advantech PCL-721 tipusú kártyával monitorozunk.

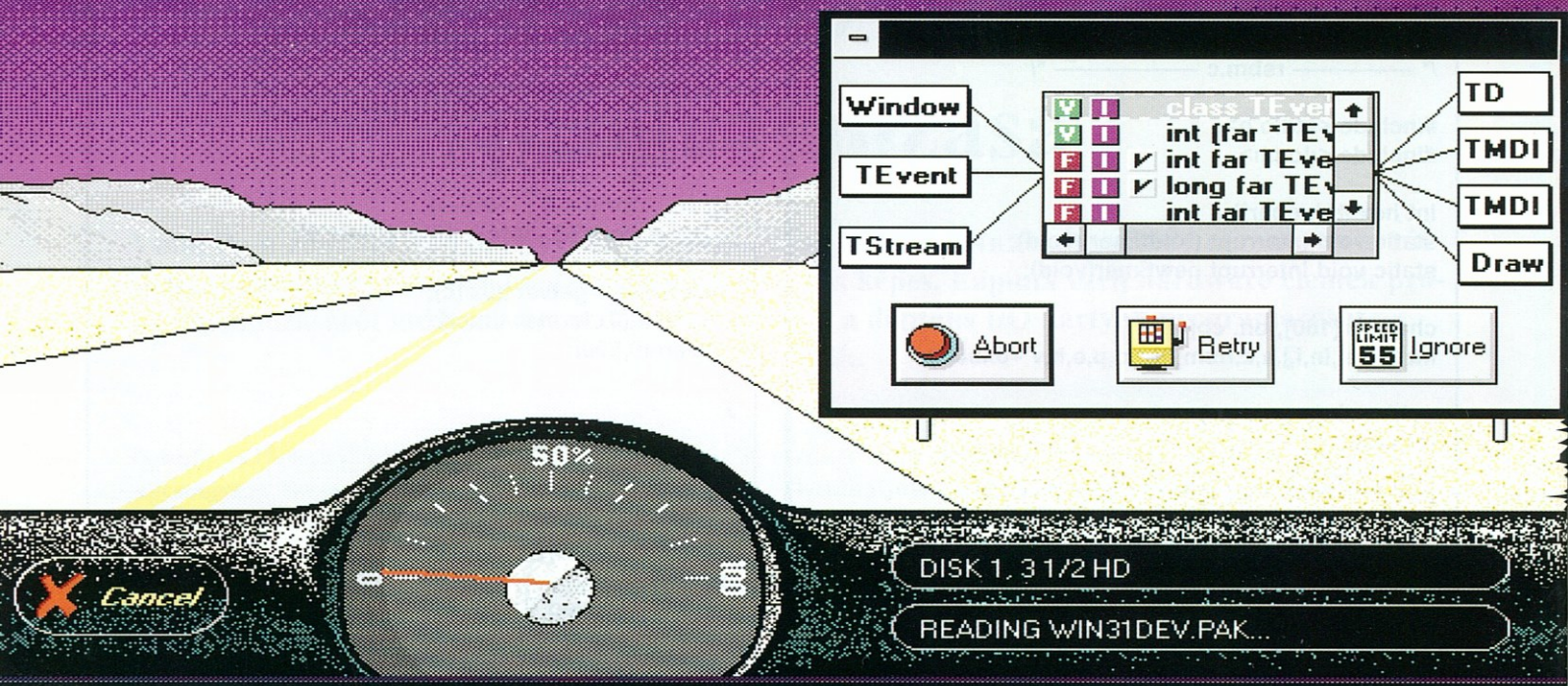
Báziscím	= 200h
Initport	= Báziscím + 6 ,
paraméter	= 0
Csatorna szám	= 32

Parancssor : RSBM 200 4 1 6 0 116 31 PCL - 721

Az eredmény a képernyő tetején jelentkezik. Az inaktív csatornák kék alapon szürke, az aktív csatornák fehér alapon piros színnel jelennek meg.

```
/* Rezidens egysoros bitmonitor  */
/*                               */
/* ————————— rsbm.c ————————— */

#include <stdio.h>
#include <dos.h>

int hexatoi (char*);
static void interrupt (*oldtimer)(void);
static void interrupt newtimer(void);

char sor[160], bit, cb[5];
int basel,ln,i,j,a,c,num[8],r,o,p,e,h,v =0xb800;

void main(argc,argv)
int argc;
char *argv[ ];
{
if       (argc == 1)
 {
 puts (" REZIDENS EGYSOROS BITMONITOR");
 puts (" ==============================");
 puts ("Indítás: \n");
 puts ("RSBM [startcím] [portszám] [sor] [initport]
        [initparam] [szín1] [szín2] [cimke]\n");
 puts ("startcim  = elsõ port címe hexadecimálisan ");
 puts ("portszám= az egy sorban kijelzett 8 bites por-
        tok száma  /max = 8/ ");
 puts ("sor         =  a képernyõ hányadik sorában
                       történik a kijelzés");
 puts (" initport   =  az inicializáló port relatív
                       távolsága startcímtõl");
 puts ("initparam =    az inicializáláshoz az initportra
                       írandó paraméter");
 puts ("szin1       =    inaktívszin ");
 puts ("szin2       =    aktívszin ");
 puts ("cimke       =    a sorban feliratot
                         helyezhetünk el\n");
 puts ("Paramétert elhagyni csak a végérõl lehet. ");
 puts ("A gép teljesítményétõl, a rendelkezésre álló
        memóriától függõen ");
 puts ("egyszerre több példányban és különbözõ
        paraméterezéssel is futhat.");
 puts ("Futásakor 10 Kb. memóriát használ
                  példányonként.");
 }

if(argc >= 2)
 {
  basel = hexatoi(argv[1]);
  if       (argc >= 3) c = atoi (argv[2]);
  else      c = 4;  if(c>8) c = 8;
  if       (argc >= 4) ln = atoi (argv[3]);
  else      ln = 1;
  if       (argc >= 6) { o = atoi (argv[4]);
                         p = atoi (argv[5]);}
  if       (argc >= 8) { e = atoi (argv[6]);
                         h = atoi (argv[7]);}
  else    { e =23; h =116;}

  for (i = 0 ; i<=80 ;i++)
          { sor [2*i]  = ' ';  sor [2*i +1] = h;}
  sprintf(cb,"%04X",(basel));
   for (i = 0 ; i<4; i++)   sor[2*i +2] = cb[i];
   i = 0;
   if (argc >=9) while(argv[8][i])
          { sor [2*i+12] = argv[8][i]; i++;}
   for (j = 0 ; j<=(c-1) ;j++)
          {
          for (i = 8 ; i>0 ;i—)
          sor[158 -18*c + 18*j+2*i]=i+'0';
          }
   oldtimer = getvect(0x1c);
   setvect(0x1c, newtimer);
   keep (0,600);
 }
}


int hexatoi(char *szoveg )
{
int szam = 0;
int i = 0;
for(i=0;  (szoveg[i]>='0' && szoveg[i]<='9') ||
           (szoveg[i]>='a' && szoveg[i]<='f') ||
           (szoveg[i]>='A' && szoveg[i]<='F')  ;    i++)
{
 if  (szoveg[i]>='0' && szoveg[i]<='9')szam = 16*szam
+ szoveg[i] -'0' ;
 if  (szoveg[i]>='a' && szoveg[i]<='f')szam = 16*szam
+ szoveg[i] -'a'+ 10 ;
 if  (szoveg[i]>='A' && szoveg[i]<='F')szam = 16*szam
+ szoveg[i] -'A'+ 10 ;
}
return(szam);
}



static void interrupt newtimer()
{
   (*oldtimer)();
   if (r == 0)
   {
      r = 1;
      if (o) outportb(basel + o , p);
      for (j = 0 ; j<=(c-1) ;j++)
          {
                num[j] = inportb(basel+j);
                num[j]  = num[j]<<8;
                 for (i = 8 ; i>0 ;i—)
                 {
                  bit = !(!(num[j] & 0x8000));
                  a =( bit ? e : h);
                  sor[159 - 18*c + 18*j+2*i]=a;
                  num[j] = num[j] <<1;
                 }
                for (i = 0 ; i<160; i++)
                pokeb(v,(ln-1)*160+i,sor[i]);
          r = 0;
          }
   }
}
```

Borland C++



Ebben a cikkben a BORLAND cég legújabb C++ fordítójának telepítésével, használatával kapcsolatos első élményeimet mesélem el.

Valódi nehézbombázó, nagyon nehezen emelkedik fel. A hozzám került 4.02 UPDATE verzió 20 lemezének telepítése után kb. 65 Mbyte nagyságú alkönyvtár jött létre, és a Windows is nehezebb lett, néhány Mbyte-tal. A könvtárszerkezet egyszerű, átlátható nem csinál olyan rendetlenséget mint az elődje, bár a WINDOWS\BORLAND.TEMP alkönyvtárban benne felejti a telepítő file-okat.
Első meglepetések: - a telepítő csak Windows alatt indítható
- az integrált környezet csak Windows alatt működik

Barátkozzunk! Először ahogyan a prospektus ajánlja Windows alól fejlesztek DOS alkalmazást, elindítom a HELLO.IDE projectfile-t. A megváltozott kiterjesztésű bináris projectfile a klasszikus hello.c forrásfile-t fordítja le.

1. LISTA

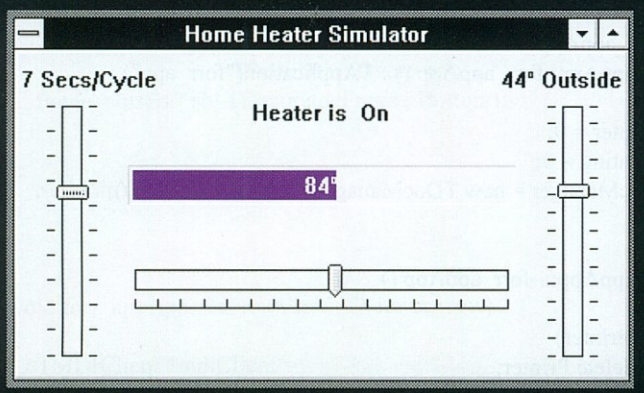
```
/*          HELLO.C -- Hello, world */
#include <stdio.h>
int main()
        {
        printf("Hello, world\n");
        return 0;
        }
```

Az eredmény: 39 092 Byte méretű hello.exe, ez többet nem fog előfordulni.Ugyan ezen file a rendszer parancssoros BCC.EXE v 4.02 fordítójával 8878 Byte, míg a nagy elöd a TCC.EXE v 2.0 -val 6544 Byte méretű futtatható filet készít. Az előzetes ismerkedés után valami életszagú feladatot nézek. Lefordítom a Quinn Curtis Inc. "Real-Time Graphics & Measurement Control Tools" című rutinkönyvtárát ami 32 forrásnyelvű file-t tartalmaz 505 583 byte terjedelemben, majd ennek felhasználásával egy példaprogramot. A műveletet összehasonlítás végett elvégeztem a fordító előző változataival is, a futás sebességét a példaprogramban lévő sebességteszt alapján próbálom összemérni.A fordítást 386 DX40 processzorú és 12 msec elérési idejű IDE merevlemezzel felszerelt gépen végeztem ,az eredményeket táblázatba foglaltam:

Fordító	:	BCC 4.0	BCC 3.1	TCC 2.0
LIB file mérete /byte/	:	204 288	206 333	208 896
DEMO file mérete /byte/	:	263 060	244 552	253 228
LIB fordítási ideje /sec/	:	111.5	95.1	45,7

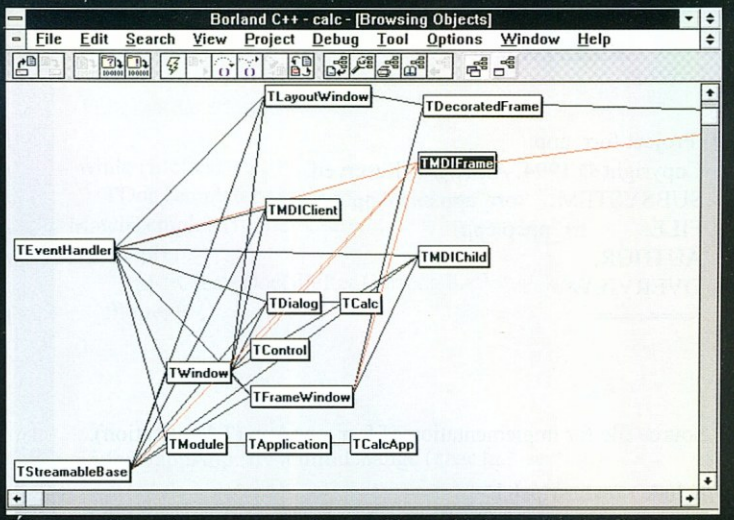
A futási sebességet mindhárom esetben közel azonosnak találtam. Összegezve a parancssoros fordító az előző változathoz hasonlóan működik, sebessége kissé elmarad attól. DOS programfejlesztők ne használják az integrált környezetet egy jó editorral néhány BAT file-val sokkal messzebb jutnak.
Következő próba a hello.c Windows-os megfelelőjének a hello.ccp-nek a fordítása. Az eredmény hasonló IDE alól fordítva 26 679 byte makefile segítségével 8360 byte az EXE file mérete. Újabb próbálkozás egy egyszerű rajzolóprogram owlpaint.cpp IDE alól fordítva 388 071 byte makefilével 57 520 bytos futtatható file-t készít. Tanulság: Windows alá se fordítsunk az integrált környezetből. Ezután nézzük a rendszer újdonságát a kódgenerátort. Az AppExpert névre hallgató csoda az IDE alatt menűből érhető el. Segítségével némi egértologatás árán dialógus dobozok kérdéseire válaszolva állíthatunk elő egyszerűbb alkalmazásokat vagy bonyolultabbak vázát, részleteit. Ráadásul mindent forrásban a definíciós és erőforrás file-okkal, makefile-vel együtt. Mindezt rém egyszerűen, bájosan, érthetően. Bár engem a konyhai robotgépre emlékeztet, amivel némi rámolás, szerelés árán megdarálhatom a kávémat, egy a kávédarálónál tízszer nagyobb tízszer drágább masinával ami tízszer annyi energiát fogyaszt, de irigyel érte a szomszéd mert olyan jól néz ki. Első próbálkozásként néhány kattintás után egy üres alkönyvtárban a 2. lista szerinti file-ok jöttek létre:

Fordítás után 794 721 byte méretű több ablakos editort kapunk eszközsávval, nyomtatás előtti megtekités lehetőségével, minden divatos jóval. Ami igazán érdekes: hogyan programoz a gép. A 3. listán a főprogramot tartalmazó file-t láthatjuk. Az integrált környezet a már elkészült model vizsgálatához, finomításához nagyon sok és nagyon látványos segítséget nyújt. Leglátványosabbak az osztályok kapcsolatait, hierarchiáját változóikat megjelenítő táblák.

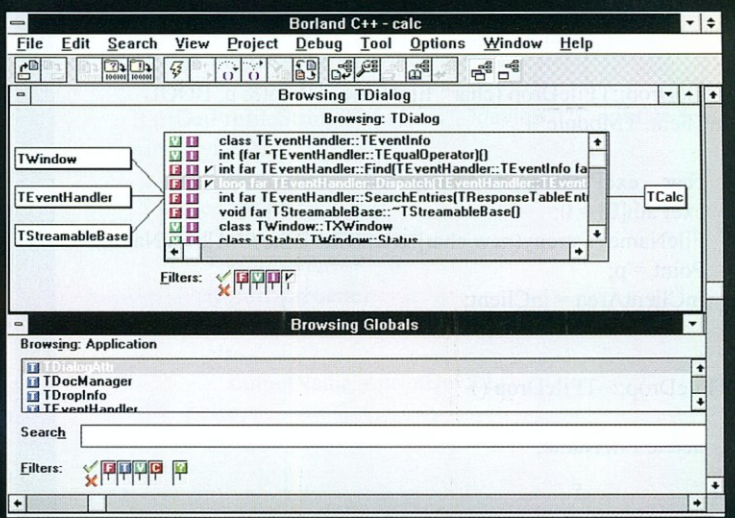


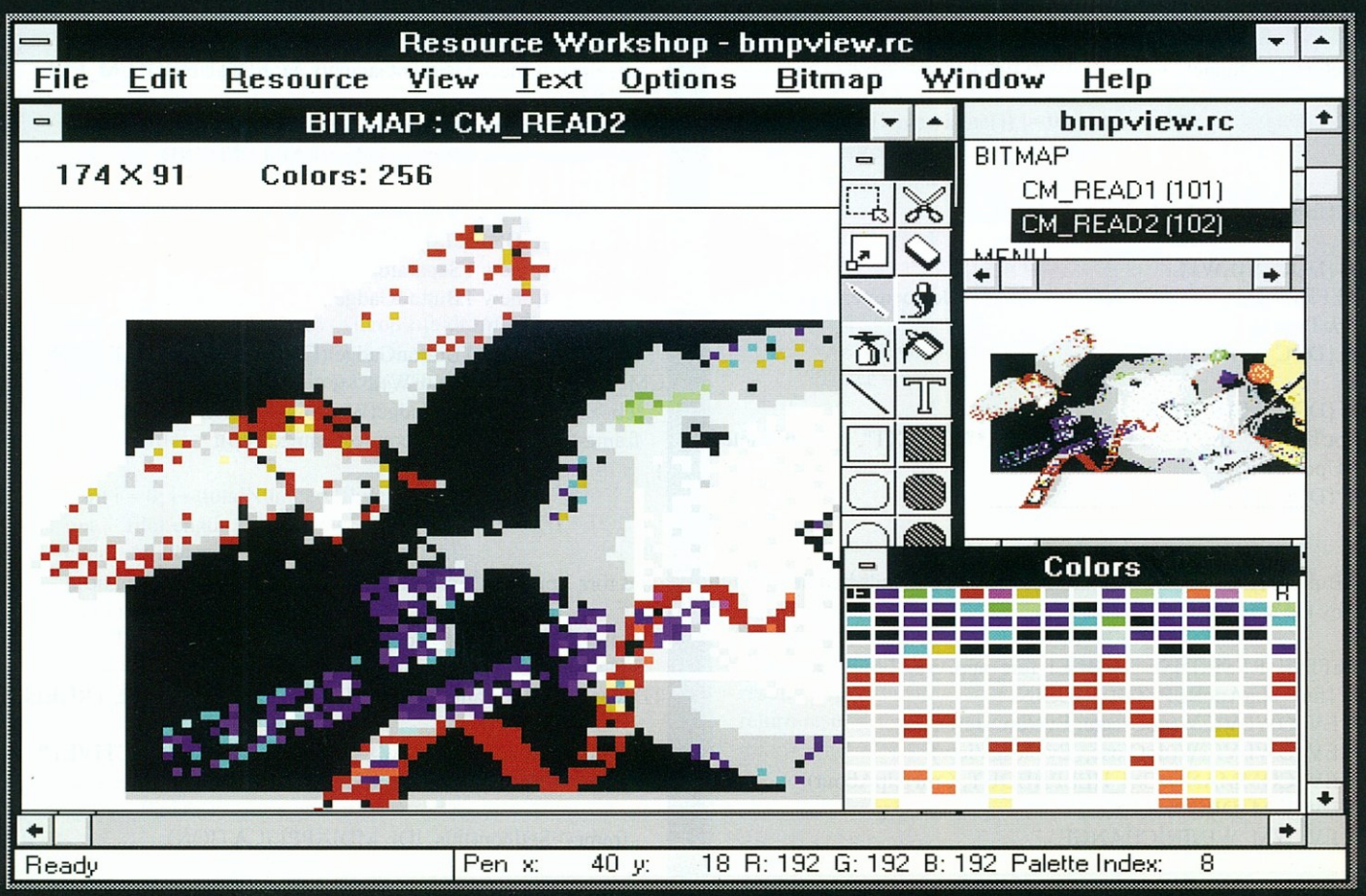
Bővült az Object Windows több új osztály, új kezelőszervek, szebb ablakok. Az erőforrás szerkesztő első látásra nem sokat változott. A kezelés komfortja megmaradt, de nem mindenben kompatibilis az elődjével. A jelenlegi változat menüi dialógus dobozai logikusabbak, kézhez állóbbak. Akinek megvan az előző változat ne dobja el, mert az azzal készült erőforrással szerkesztett EXE file-t az új nem mindig tudja megnyitni.
Az új kezelőszervek használatának lehetősége, a szebb megjelenítés, az erőforrások egyértelműbb konfigurálási lehetősége miatt azonban az új használatát javaslom.

Az új erőforrás szerkesztő akár rajzoló programként is megállná a helyét.



Újra tanulhatunk programozni,a grafikus objektumokkal való munka ellentétben a reklámszlogennel nem kezdőknekvaló.Nagy méretű alkalmazásokat fejleszthetnek vele a profik hatékonyabban, biztonságosabban mint a hagyományos módon.



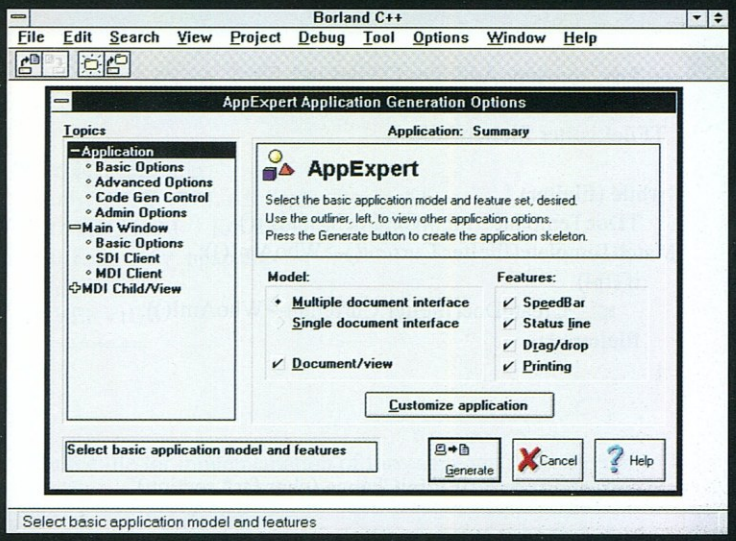


3. LISTA

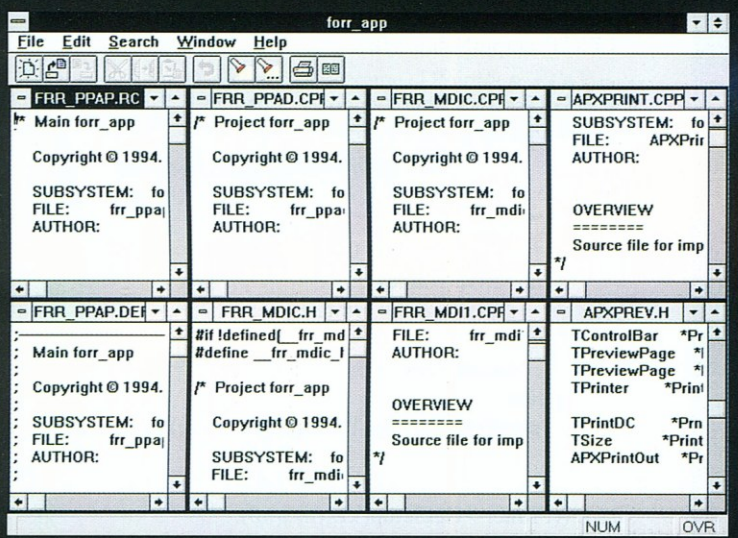
```
/*  Project forr_app
    Copyright © 1994. All Rights Reserved.
    SUBSYSTEM:    forr_app.exe Application
    FILE:         frr_ppap.cpp
    AUTHOR:
    OVERVIEW
    ========



  Source file for implementation of forr_appApp (TApplication).
*/
#include <owl\owlpch.h>
#pragma hdrstop
#include <dir.h>
#include "frr_ppap.h"
#include "frr_mdic.h"
#include "frr_mdi1.h"
#include "frr_ppad.h"

TFileDrop::TFileDrop (char* fileName, TPoint& p, BOOL
inClient, TModule*)
{
   char    exePath[MAXPATH];
   exePath[0] = 0;
   FileName = strcpy(new char[strlen(fileName) + 1], fileName);
   Point = p;
   InClientArea = inClient;
}

TFileDrop::~TFileDrop ()
{
   delete FileName;
}

const char *TFileDrop::WhoAmI ()
{
 return FileName;
}



//{{forr_appApp Implementation}}

//{{DOC_VIEW}}
DEFINE_DOC_TEMPLATE_CLASS(TFileDocument, TEditView,
DocType1);
//{{DOC_VIEW_END}}

//{{DOC_MANAGER}}
DocType1 __dvt1("All Files (*.*)", "*.*", 0, "TXT", dtAutoDelete |
dtUpdateDir);
//{{DOC_MANAGER_END}}

//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(forr_appApp, TApplication)
//{{forr_appAppRSP_TBL_BEGIN}}
   EV_OWLVIEW(dnCreate, EvNewView),
   EV_OWLVIEW(dnClose,  EvCloseView),
   EV_COMMAND(CM_HELPABOUT, CmHelpAbout),
   EV_WM_DROPFILES,
   EV_WM_WININICHANGE,
//{{forr_appAppRSP_TBL_END}}
END_RESPONSE_TABLE;


//////////////////////////////////////////////////////
// forr_appApp
// ======
//
//
// forr_appApp
forr_appApp::forr_appApp () : TApplication("forr_app")
{
   Printer = 0;
   Printing = 0;
   DocManager = new TDocManager(dmMDI | dmMenu);
}

forr_appApp::~forr_appApp ()
{
   if (Printer)
      delete Printer;
}


void forr_appApp::SetupSpeedBar (TDecoratedMDIFrame *frame)
{
     TControlBar* cb = new TControlBar(frame);
   cb->Insert(*new TButtonGadget(CM_MDIFILENEW,
CM_MDIFILENEW));
   cb->Insert(*new TButtonGadget(CM_MDIFILEOPEN,
CM_MDIFILEOPEN));
   cb->Insert(*new TButtonGadget(CM_FILESAVE, CM_FILE-
SAVE));
   cb->Insert(*new TSeparatorGadget(6));
   cb->Insert(*new TButtonGadget(CM_EDITCUT, CM_EDIT-
CUT));
   cb->Insert(*new TButtonGadget(CM_EDITCOPY, CM_EDIT-
COPY));
   cb->Insert(*new TButtonGadget(CM_EDITPASTE, CM_EDIT-
PASTE));
   cb->Insert(*new TSeparatorGadget(6));
   cb->Insert(*new TButtonGadget(CM_EDITUNDO, CM_EDI-
TUNDO));
   cb->Insert(*new TSeparatorGadget(6));
   cb->Insert(*new TButtonGadget(CM_EDITFIND,
CM_EDITFIND));
   cb->Insert(*new TButtonGadget(CM_EDITFINDNEXT,
CM_EDITFINDNEXT));
   cb->Insert(*new TSeparatorGadget(6));
   cb->Insert(*new TButtonGadget(CM_FILEPRINT,
CM_FILEPRINT));
   cb->Insert(*new TButtonGadget(CM_FILEPRINTPREVIEW,
CM_FILEPRINTPREVIEW));
    cb->SetHintMode(TGadgetWindow::EnterHints);
   frame->Insert(*cb, TDecoratedFrame::Top);
}



void forr_appApp::InitMainWindow ()
{
   mdiClient = new forr_appMDIClient;
   TDecoratedMDIFrame* frame = new
TDecoratedMDIFrame(Name, MDI_MENU, *mdiClient, TRUE);

   nCmdShow = (nCmdShow != SW_SHOWMINNOACTIVE) ?
SW_SHOWNORMAL : nCmdShow;

    frame->SetIcon(this, IDI_MDIAPPLICATION);
```

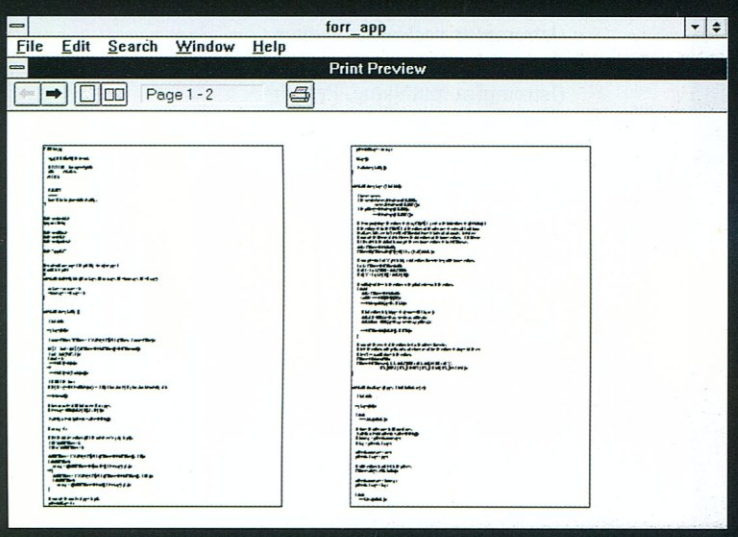
```
  frame->AssignMenu(MDI_MENU);
    frame->Attr.AccelTable = MDI_MENU;


  SetupSpeedBar(frame);

  TStatusBar *sb = new TStatusBar(frame, TGadget::Recessed,
                          TStatusBar::CapsLock      |
                          TStatusBar::NumLock       |
                          TStatusBar::ScrollLock    |
                          TStatusBar::Overtype);
  frame->Insert(*sb, TDecoratedFrame::Bottom);


  SetMainWindow(frame);

}

void forr_appApp::EvNewView (TView& view)
{
  TMDIClient *mdiClient =
TYPESAFE_DOWNCAST(GetMainWindow()-
>GetClientWindow(), TMDIClient);
  if (mdiClient) {
    forr_appMDIChild* child = new
forr_appMDIChild(*mdiClient, 0, view.GetWindow());

    // Associate ICON w/ this child window.
    child->SetIcon(this, IDI_DOC);

    child->Create();
  }
}


void forr_appApp::EvCloseView (TView&)
{
}

void forr_appApp::CmHelpAbout ()
{
    forr_appAboutDlg(MainWindow).Execute();
}

void forr_appApp::InitInstance ()
{
  TApplication::InitInstance();
  GetMainWindow()->DragAcceptFiles(TRUE);
}

void forr_appApp::EvDropFiles (TDropInfo drop)
{
    int totalNumberOfFiles = drop.DragQueryFileCount();

  TFileList* files = new TFileList;

  for (int i = 0; i < totalNumberOfFiles; i++) {
     int    fileLength = drop.DragQueryFileNameLen(i) + 1;
    char   *fileName = new char[fileLength];
    drop.DragQueryFile(i, fileName, fileLength);
    TPoint  point;
    BOOL   inClientArea = drop.DragQueryPoint(point);
    files->Add(new TFileDrop(fileName, point, inClientArea,
this));
  }

  AddFiles(files);
  drop.DragFinish();
}
```

```
void forr_appApp::AddFiles (TFileList* files)
{
  TFileListIter fileIter(*files);

  while (fileIter) {
    TDocTemplate* tpl = GetDocManager()-
>MatchTemplate(fileIter.Current()->WhoAmI());
    if (tpl)
      tpl->CreateDoc(fileIter.Current()->WhoAmI());
    fileIter++;
  }
}


void forr_appApp::EvWinIniChange (char far* section)
{
  if (lstrcmp(section, "windows") == 0) {
       if (Printer) {
       char printDBuffer[255];
       LPSTR printDevice = printDBuffer;
       LPSTR devName = 0;
       LPSTR driverName = 0;
       LPSTR outputName = 0;

       if (::GetProfileString("windows", "device", "", printDevice,
sizeof(printDevice))) {
                devName = printDevice;
          while (*printDevice) {
            if (*printDevice == ',') {
              *printDevice++ = 0;
              if (!driverName)
                driverName = printDevice;
              else
                outputName = printDevice;
            } else
              printDevice = AnsiNext(printDevice);
          }

          if ((Printer->GetSetup().Error != 0)                    ||
            (lstrcmp(devName, Printer-
>GetSetup().GetDeviceName()) != 0)    ||
            (lstrcmp(driverName, Printer-
>GetSetup().GetDriverName()) != 0) ||
            (lstrcmp(outputName, Printer-
>GetSetup().GetOutputName()) != 0)) {

            delete Printer;
            Printer = new TPrinter;
          }
       } else {
              delete Printer;
          Printer = new TPrinter;
       }
    }
  }
}


int OwlMain (int , char* [])
{
  forr_appApp    App;
  int            result;
  result = App.Run();
  return result;
}
```



Az igazi újdonság a kódgenerátor ablaka és az általa létrehozott alkalmazás képei.





Ahogyan az már lenni szokott vizsgálódásaim közben sok hibát is találtam. A DPMI loader nem szereti ha a memóriában rezidens programok csücsülnek. A példaprogramok némelyike nem fordítható.A fordító néha elfelejti visszakapcsolni a processzort, és a linker nem.fut le.

Ezekről és még más tapasztalatokról a következő számokban olvashatnak bővebben lesz szó. Búcsúzóul az egyik példaprogram képernyője.

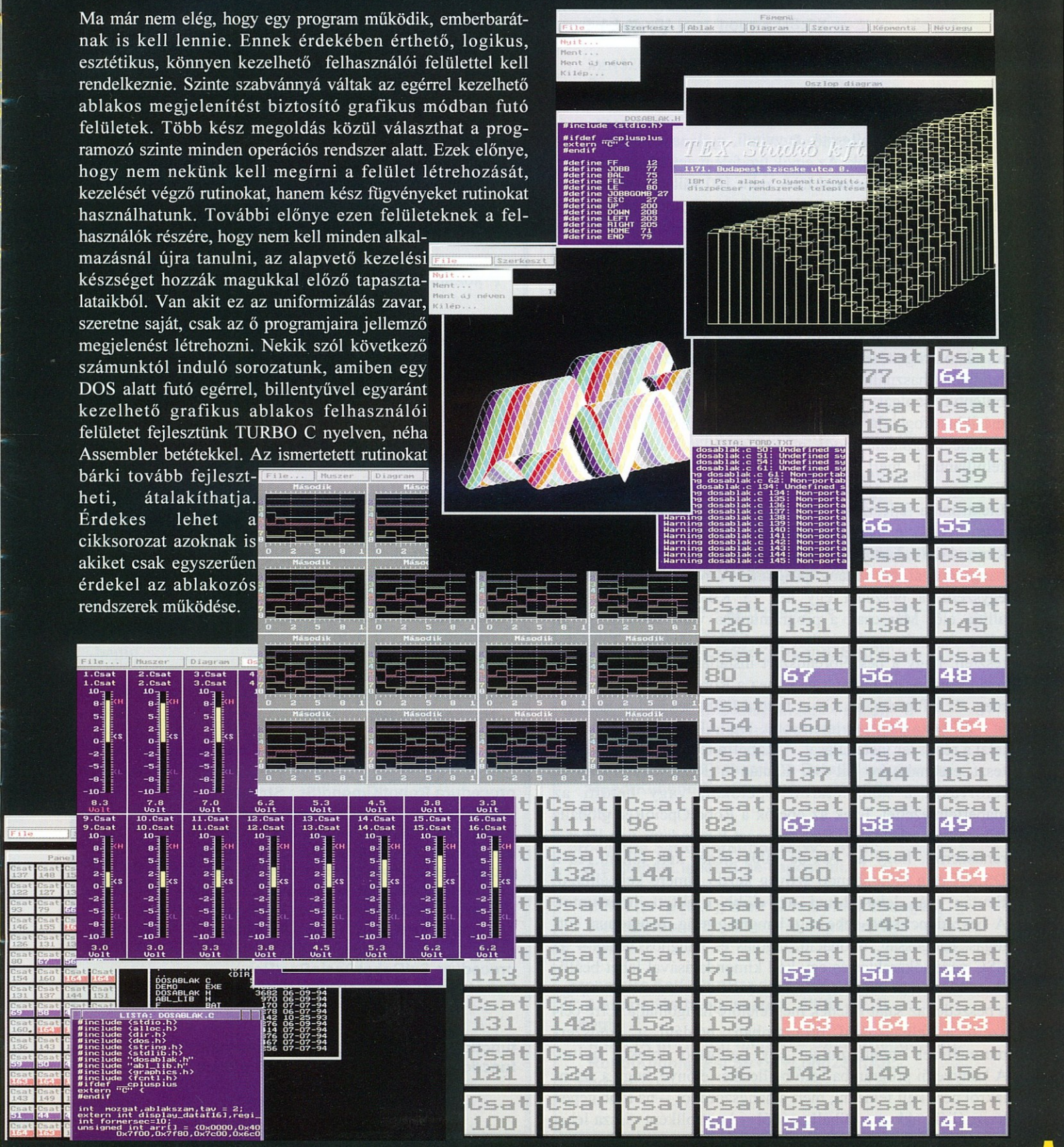
Nagy Sándor

2. LISTA

FORR_APP APX	36351	09-21-94 4:57p
FRR_PPAP RC	17092	09-21-94 4:56p
FRR_PPAP RH	6366	09-21-94 4:57p
FRR_PPAP DEF	517	09-21-94 4:57p
FRR_PPAP CPP	9426	09-21-94 4:57p
FRR_PPAP H	2559	09-21-94 4:56p
APXPREV CPP	9803	09-21-94 4:57p
APXPREV H	1888	09-21-94 4:57p
FRR_PPAD CPP	4972	09-21-94 4:57p
FRR_PPAD H	1500	09-21-94 4:57p
FRR_MDI1 CPP	5159	09-21-94 4:57p
FRR_MDI1 H	1251	09-21-94 4:57p
FRR_MDIC CPP	5057	09-21-94 4:57p
FRR_MDIC H	1407	09-21-94 4:57p
APXPRINT CPP	5621	09-21-94 4:57p
APXPRINT H	1289	09-21-94 4:57p
NEW BMP	358	09-21-94 4:57p
OPEN BMP	358	09-21-94 4:57p
SAVE BMP	358	09-21-94 4:57p
UNDO BMP	358	09-21-94 4:57p
CUT BMP	358	09-21-94 4:57p
COPY BMP	358	09-21-94 4:57p
PASTE BMP	358	09-21-94 4:57p
FIND BMP	358	09-21-94 4:57p
FINDNEXT BMP	358	09-21-94 4:57p
PREVIEW BMP	358	09-21-94 4:57p
PRINT BMP	358	09-21-94 4:57p
PREVIOUS BMP	322	09-21-94 4:57p
NEXT BMP	322	09-21-94 4 :57p
PREVIEW1 BMP	322	09-21-94 4:57p
APPLDOCV ICO	1086	09-21-94 4:57p
MDICHILD ICO	1086	09-21-94 4:57p
FORRAS MAK	2547	09-21-94 4:57p
FORRAS IDE	27610	09-21-94 4:57p
FORRAS DSW	528	09-21-94 4:57p

Az Ön programjának van saját arca?

Ma már nem elég, hogy egy program működik, emberbarátnak is kell lennie. Ennek érdekében érthető, logikus, esztétikus, könnyen kezelhető felhasználói felülettel kell rendelkeznie. Szinte szabvánnyá váltak az egérrel kezelhető ablakos megjelenítést biztosító grafikus módban futó felületek. Több kész megoldás közül választhat a programozó szinte minden operációs rendszer alatt. Ezek előnye, hogy nem nekünk kell megírni a felület létrehozását, kezelését végző rutinokat, hanem kész függvényeket rutinokat használhatunk. További előnye ezen felületeknek a felhasználók részére, hogy nem kell minden alkalmazásnál újra tanulni, az alapvető kezelési készséget hozzák magukkal előző tapasztalataikból. Van akit ez az uniformizálás zavar, szeretne saját, csak az ő programjaira jellemző megjelenést létrehozni. Nekik szól következő számunktól induló sorozatunk, amiben egy DOS alatt futó egérrel, billentyűvel egyaránt kezelhető grafikus ablakos felhasználói felületet fejlesztünk TURBO C nyelven, néha Assembler betétekkel. Az ismertetett rutinokat bárki tovább fejlesztheti, átalakíthatja. Érdekes lehet a cikksorozat azoknak is akiket csak egyszerűen érdekel az ablakozós rendszerek működése.



Visszabeszél a könyvelőprogram

Lapunkban rendszeresen megszólaltatunk gyakorló profikat. Az első számban az apropót egy könyvelőprogramban alkalmazott érdekes megoldás adta.

Én, mint szoftverfejlesztő, ügyviteli, nyilvántartási és vezérlési programokat írok. Az egyik ügyfelem egy könyvelési cég, amely kevés alkalmazottal több, mint 100 vállalkozás bérszámfejtését és könyvelését végzi. Egy ilyen cégnél nagyon fontos az időtényező. A sebességet a számítógépekkel és a programokkal bizonyos határok között ugyan lehet növelni, de az adatok rögzítése nagyságrendekkel több időt visz el, mint a feldolgozás. Jelentősen csak a rögzítés gyorsításával javíthatjuk a teljesítményt. Hogyan valósítható ez meg?
Először is mivel a könyvelő egyik keze foglalt a programban az adatok rögzítéséhez elég csak a keypadot a numerikus billennyűket használni. A mentésre a *, egy lista lekérésére a + billentyű szolgál, így a rögzítő egyik keze felszabadul, a másikban foghatja a rögzítésre váró számlákat. Már csak azt kellene megoldani, hogy ne kelljen állandóan a monitorra felpillantani, mert hát ugye az ellenőrzés nagyon fontos. Erre egy remek, egyszerű és nem túl drága megoldás, a Nikol Elektronika fejlesztése a PC-ROBOT kínálkozik. Ez a rendszer lehetővé teszi a magyar nyelvű szöveg érthető kimondását. Tartozik hozzá egy beszédszintetizátorkártya hangszóróval és néhány program. A rendszer felélesztése egyszerű, a beszédet egy kb. 75 Kb-os rezidensprogram állítja elő. Már csak a programunkba való beillesztés van hátra. Az unit meghívásakor a RobotInit eljáráson keresztül teszteli, hogy a beszélőrendszert installálták-e, a szükséges rezidensprogram elindult-e. Ehhez a RobotInitedProcot hívja meg. Ez a fügvény az AX-et feltölti F000 hexával és meghívja a 2F hexa szoftver interruptot, ami AL = FF hexaértékkel jelzi a rendszer installálását. A RobotInited globális változó, a teszt eredményére áll be. Ezután a ReadRobotOpcioRec felolvassa a már letárolt opciókat, vagy betölti az alapértelmezettet. Ezek az opciók a RobotOpcioRec globális változóban érhetők el. Egy szöveget kimondatni úgy lehet a rendszerrel, hogy az AX-be F001 hexát kell tölteni, és a DS:DX-et a kimondandó szöveg kezdő címére kell állítani. Ezután már lehet hívni az előbb említett interruptot. A beszédopcióit -- sebesség, hangmagasság, hangtónus, suttogás, hangosság, intonáció, tagolás és hanghosszúság -- az ESC vezérlőkarakter és egy kód kimondatásával lehet beállítani. Az opciók beállítását célszerű a felhasználóra bízni, így a beszédet az saját fülének megfelelőre hangolhatja. Erre célszerű a felhasználói programban, például a rendszerparaméterek menüpontban egy beolvasó ablakot nyitni. A könyvelői programba való beillesztés előnye, hogy a számlákról nem kell felnézni, mert a hangszóróból -- ha több gépen használják, akkor a fülhallgatóból -- kontrollként vissza lehet hallani a begépelt vagy kiválasztott adatokat. Pl: „Kiss István tartozik Tízezer forint + huszonöt százalék áfával" Ezen a megoldáson kívül még számtalan helyen fel lehet használni a megadott egyszerű uniton -- RobotDrv.Pas -- keresztül a PC-ROBOT beszélőrendszert.

Balla László vill. mérnök
Telefon: 226-8962

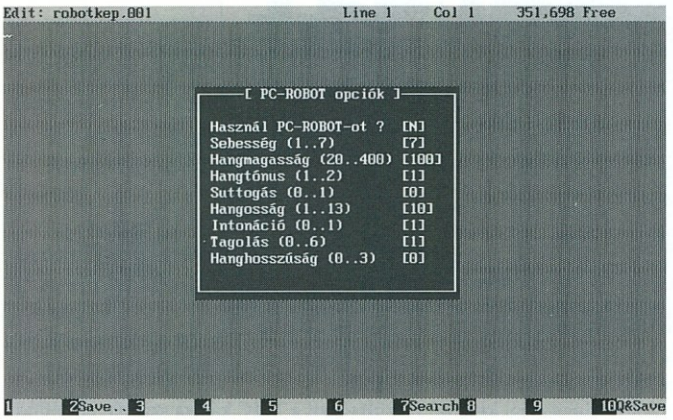
1. Lista **ROBOTDRV.PAS**

```
unit RobotDrv;

interface  type

(* A robot opció rekord tipus deklarációja *)

  RobotOpcioRecTipus = record
    Hasznalatban     : Boolean;
    Sebesseg         : Byte;
    Hangmagassag     : Word;
    Hangtonus        : Byte;
    Suttogas         : Byte;
    Hangossag        : Byte;
    Intonacio        : Byte;
    Tagolas          : Byte;
    Hanghosszusag    : Byte;
  end;


procedure RobotKimond(St : String);
procedure SetRobotSebesseg(S : Byte);
procedure SetRobotHangmagassag(P : Word);
procedure SetRobotHangtonus(V : Byte);
procedure SetRobotSuttogas(W : Byte);
procedure SetRobotHangossag(A : Byte);
procedure SetRobotIntonacio(I : Byte);
procedure SetRobotTagolas(X : Byte);
procedure SetRobotHanghosszusag(T : Byte);
procedure SetRobotOpcioRec(Rec :
RobotOpcioRecTipus);


function WriteRobotOpcioRec : Boolean;
function ReadRobotOpcioRec : Boolean;
```

```
var

(* Globális változók *)

  RobotInited   : Boolean;
                  (* A rendszer inicializálva van-e?)
  RobotOpcioRec : RobotOpcioRecTipus;
                  (* Az aktuális opciókat tárolórekord*)
implementation
uses
 Dos;

const
  EscStr            = #27;
  RobotFName        = 'ROBOT.DAT';
( Annak file-nak a neve, amiben az opciókat tároljuk*)
DefaultSebesseg          = 7;
DefaultHangmagassag   = 100;
DefaultHangtonus         = 1;
DefaultSuttogas          = 0;
                  (* Az alapértelmezett beállítások *)
  DefaultHangossag       = 10;
  DefaultIntonacio    =   1;
  DefaultTagolas      =   1;
  DefaultHanghosszusag = 0;

procedure ErrorMessage(St : String);
                          (* Hibaüzenet kiírása. *)
        (* Az aktuális környezetben ezt az eljárást kell
                                        kicserélni.
*)
  begin
    WriteLn(St);
  end;

function Long2Str(L : LongInt) : String;
                  (* Egész sztringgé alakítása. *)
  var
    St : String;
  begin
    Str(L, St);
    Long2Str := St;
  end;

function RobotInitedProc : Boolean;       (* A beszélő
rendszer installálását teszteli. *)
  var
    R : Registers;
    S : Word;
  begin
    R.AX := $F000;
    Intr($2F, R);
(* 2F hexa szoftver interrupton keresztül lehet elérni.*)
    RobotInitedProc := (R.AL = $FF);
  end;

        (* Ha a visszatérési érték FF hexa, a *)
        (* rendszer installáva van. *)

procedure LowRobotKimond(St : String);

        ( * A rendszer tesztelése nélkül kimondja
              a megkapott St sztringet. *)
```

```
 var
    R : Registers;
    S : Word;
  begin
    R.DS := Seg(St);
    R.DX := Ofs(St);
    R.AX := $F001;
   (* Az AL=01 jelzi, hogy a DS:DX az St kezdőcímére
                                                mutat *)
    Intr($2F, R);
    S := R.AX;
    if S > 0
      then begin
        ErrorMessage      ('PC Robot hiba :
'+Long2Str(S));
        exit;
      end;
  end;

procedure RobotKimond(St : String);
          (* Ha a rendszer installálva van, a megkapott
                  St sztringet átadja kimondásra. *)
  begin
    if not RobotInited then exit;
    LowRobotKimond(St);
  end;

procedure SetRobotSebesseg(S : Byte);
          (* A kimondás sebességét  állítja be. *)
  begin
    if not (S in [1..7]) then S := DefaultSebesseg;
    RobotKimond(EscStr+'s'+Long2Str(S));
  end;

procedure SetRobotHangmagassag(P : Word);
(* A kimondás hangmagasságát  állítja be. *)
  begin
    if (P < 20) or (P > 400) then P :=
DefaultHangmagassag;
    RobotKimond(EscStr+'p'+Long2Str(P));
  end;

procedure SetRobotHangtonus(V : Byte);
          (* A kimondás hangtónusát állítja be. *)
  begin
    if not (V in [1..2]) then V := DefaultHangtonus;
    RobotKimond(EscStr+'v'+Long2Str(V));
  end;

procedure SetRobotSuttogas(W : Byte);
(* A suttogást lehet ki-be kapcsolni. *)
  begin
    if not (W in [0..1]) then W := DefaultSuttogas;
    RobotKimond(EscStr+'w'+Long2Str(W));
  end;

procedure SetRobotHangossag(A : Byte);
(* A kimondás hangosságát  állítja be. *)
  begin
    if not (A in [1..13]) then A := DefaultHangossag;
    RobotKimond(EscStr+'a'+Long2Str(A));
  end;
```

```
procedure SetRobotIntonacio(I : Byte);
(* A kimondás intonációját állítja be. *)
 begin
  if not (I in [0..1]) then I := DefaultIntonacio;
  RobotKimond(EscStr+'i'+Long2Str(I));
 end;

procedure SetRobot Tagolas(X : Byte);
(* A kimondás tagolását  állítja be. *)
 begin
  if not (X in [0..6]) then X := DefaultTagolas;
  RobotKimond(EscStr+'x'+Long2Str(X));
 end;

procedure SetRobotHanghosszusag(T : Byte);
(* A kimondás hanghosszúságát állítja be. *)
 begin
  if not (T in [0..3]) then T := DefaultHanghosszusag;
  RobotKimond(EscStr+'t'+Long2Str(T));
 end;

procedure SetRobotOpcioRec(Rec :
RobotOpcioRecTipus);
(* A megkapott robot opció rekord alapján beállítja az
                                    opciókat. *)
 begin
  with Rec do
   begin
    SetRobotSebesseg(Sebesseg);
    SetRobotHangmagassag(Hangmagassag);
    SetRobotHangtonus(Hangtonus);
    SetRobotSuttogas(Suttogas);
    SetRobotHangossag(Hangossag);
    SetRobotIntonacio(Intonacio);
    SetRobotTagolas(Tagolas);
    SetRobotHanghosszusag(Hanghosszusag);
   end;
 end;

function WriteRobotOpcioRec : Boolean;
         (* Az aktuális robot opció rekordot menti le. *)
 var
  F : File of RobotOpcioRecTipus;
  I : Integer;
 begin
  WriteRobotOpcioRec := false;
  Assign(F, RobotFName);
  Rewrite(F);
  Write(F, RobotOpcioRec);
  Close(F);
  I := IOResult;
  If I <> 0 then
   begin
    ErrorMessage('Write hiba <RobotOpcioRec> :
'+Long2Str(I));
     exit;
   end;
  WriteRobotOpcioRec := true;
 end;

function ReadRobotOpcioRec : Boolean;
(* Az aktuális robot opció rekordot olvassa fel, vagy
         feltölti az alap,értelmezett adatokkal. *)
 var
  F : File of RobotOpcioRecTipus;
  I : Integer;
 begin
  ReadRobotOpcioRec := false;
  Assign(F, RobotFName);
  Reset(F);
  I := IOResult;
  case I of
   0 : begin (* Beolvassa a rekordot *)
        Read(F, RobotOpcioRec);
        Close(F);
        I := IOResult;
        if I <> 0
         then begin
          ErrorMessage('Read hiba
<RobotOpcioRec> : '+Long2Str(I));
          exit;
         end;
       end;
   2 : begin (* Feltölti a rekordot *)
        FillChar(RobotOpcioRec,
SizeOf(RobotOpcioRecTipus), 0);
        with RobotOpcioRec do
         begin
          Sebesseg := DefaultSebesseg;
          Hangmagassag := DefaultHangmagassag;
          Hangtonus := DefaultHangtonus;
          Suttogas := DefaultSuttogas;
          Hangossag := DefaultHangossag;
          Intonacio := DefaultIntonacio;
          Tagolas := DefaultTagolas;
          Hanghosszusag := DefaultHanghosszusag;
         end;
        if not WriteRobotOpcioRec then exit;
       end;
    else begin
     ErrorMessage('Read hiba <RobotOpcioRec> :
'+Long2Str(I));
      exit;
    end;
  end;
  ReadRobotOpcioRec := true;
 end;

procedure RobotInit;
         (* A unit az első meghívásakor teszteli,
         hogy a rendszer inicializálva van-e,
          és felolvassa, majd beállítja az opciókat *)
 var
  Rec : RobotOpcioRecTipus;
 begin
  RobotInited := RobotInitedProc;
  if not RobotInited then exit;
  if not ReadRobotOpcioRec then exit;
  RobotInited := RobotOpcioRec.Hasznalatban;
  if not RobotInited then exit;
  SetRobotOpcioRec(RobotOpcioRec);
 end;

begin
 RobotInit;
end.
```



1. Ábra Beállító képernyő a PC ROBOT rendszerhez

Megszakítás parancssorból

A gép lelkével most ismerkedők vehetik hasznát az alábbi három Turbo C-ben írt rutinnak. Az első listán közölt POKE.C program segítségével adott szegmens és ofszet-címre írhatunk egy tetszőleges decimális számot. A második listában lévő OUT.C a gép tetszőleges PORT-jára ír egy decimális számot. A harmadik listában lévő INT.C rutinnal tetszőleges megszakítást hívhatunk meg parancssorból. A programokat COM formátumra fordít-

2. Lista

A kipróbáláshoz egy egyszerű példa: Demo.Pas

```
program Demo;

uses
  RobotDrv;

procedure Kimond(S : String);
  begin
    RobotKimond(S+'.');
  end;

begin
  WriteLn('PC Robot');
  Kimond('PC Robot');
  WriteLn('Százhúszezerötszáz');
  Kimond('120500');
end.
```


1. Lista POKE.C

```
unsigned int seg,offs,i;
unsigned char param;

int hexatoi(char *szoveg )
{
    int szam = 0;
    int i = 0;
    for(i=0;  (szoveg[i]>='0' && szoveg[i]<='9') ||
    (szoveg[i]>='a' && szoveg[i]<='f') ||
    (szoveg[i]>='A' && szoveg[i]<='F') ;   i++)
    {
        if  (szoveg[i]>='0' && szoveg[i]<='9')
                szam = 16*szam + szoveg[i] -'0' ;
        if  (szoveg[i]>='a' && szoveg[i]<='f')
                szam =  16*szam + szoveg[i] -'a'+ 10 ;
        if  (szoveg[i]>='A' && szoveg[i]<='F')
                szam = 16*szam + szoveg[i] -'A'+ 10 ;
    } return(szam);
}
void main(argc,argv)
    int argc;
    char *argv[];
{
if(argc == 4)
    {
    seg    = hexatoi(argv[1]);
    offs   = hexatoi(argv[2]);
    for(i=0; argv[3][i]>='0' && argv[3][i]<='9';i++)
    param = 10*param + argv[3][i] -'0';
    pokeb(seg,offs,param);
    }
else
    {    puts ("Használat:  POKE  szegmenscím
                offsetcím    érték");
    puts ("        /hex/        /hex/        /dec/");
    exit(0);
    }
}
```

2. Lista. OUT.C

```
unsigned int basel,i;
unsigned char param;

void main(argc,argv)
    int argc;
    char *argv[];

{
if(argc == 3)
    {
     for(i=0;  (argv[1][i]>='0' && argv[1][i]<='9') ||
    (argv[1][i]>='a' && argv[1][i]<='f') ||
    (argv[1][i]>='A' && argv[1][i]<='F')  ;    i++)
          {
           if  (argv[1][i]>='0' && argv[1][i]<='9')
                    basel = 16*basel + argv[1][i] -'0' ;
           if  (argv[1][i]>='a' && argv[1][i]<='f')
                    basel = 16*basel + argv[1][i] -'a'+ 10 ;
           if  (argv[1][i]>='A' && argv[1][i]<='F')
                    basel = 16*basel + argv[1][i] -'A'+ 10;
          }
    for(i=0; argv[2][i]>='0' && argv[2][i]<='9';i++)
                    param = 10*param + argv[2][i] -'0';

     outportb(basel,param);
          }
    else
    puts ("Használat:  OUT   portcím     érték");
    puts ("                    /hex/      /dec/");
    }
```

3. Lista INT.C

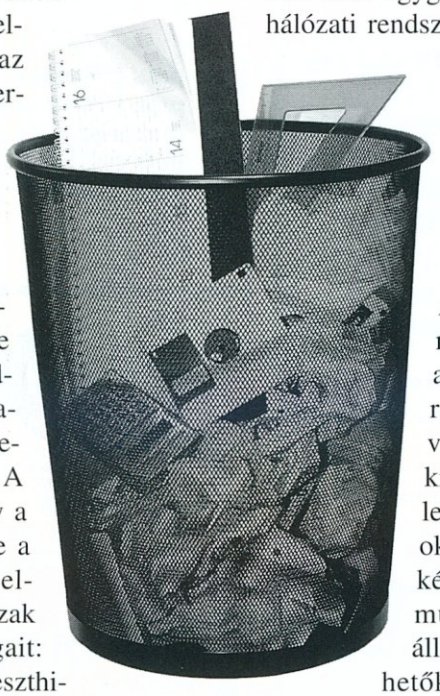
```
#include <dos.h>

int i;
unsigned int
intnum,axparam,bxparam,cxparam,dxparam;
static union REGS rg;


int hexatoi(char *szoveg )
{
int szam = 0;
int i = 0;
for(i=0;  (szoveg[i]>='0' && szoveg[i]<='9') ||
    (szoveg[i]>='a' && szoveg[i]<='f') ||
    (szoveg[i]>='A' && szoveg[i]<='F')  ;    i++)

    {
    if  (szoveg[i]>='0' && szoveg[i]<='9')
          szam =  16*szam + szoveg[i] -'0' ;
    if  (szoveg[i]>='a' && szoveg[i]<='f')
          szam =  16*szam + szoveg[i] -'a'+ 10 ;
    if  (szoveg[i]>='A' && szoveg[i]<='F')
          szam = 16*szam + szoveg[i] -'A'+ 10 ;
    }
return(szam);
}

void main(argc,argv)
    int argc;
    char *argv[];
{
if (argc >= 2)
    {
    intnum = hexatoi(argv[1]);
    if (argc >= 3){ axparam = hexatoi(argv[2]);
          rg.x.ax =  axparam;}
    if (argc >= 4){ bxparam = hexatoi(argv[3]);
          rg.x.bx =  bxparam;}
    if (argc >= 5){ cxparam = hexatoi(argv[4]);
          rg.x.cx =  cxparam;}
   if (argc == 6){ dxparam = hexatoi(argv[5]);
          rg.x.dx =  dxparam;}
   int86(intnum, &rg, &rg);
    }
else
    {
    puts ("Megszakítás parancssorból\n");
    puts ("Használat:    INT    intnum  [axparam]
    [bxparam]  [cxparam]  [dxparam]");
    puts ("                    /hex/  /hex/     /hex/
    /hex/     /hex/ ");
    }
}
```

A RENDTEREMTÉS

1
Végy egy jó (nagy) szemeteskosarat

Mi a második megoldást ajánljuk.

A számítógépek használata egyre inkább részévé válik életünknek. A legkülönbözőbb számítógépes rendszerekkel találkozhatunk már irodákban és szakrendelőkben, üzletekben és üzemekben, sőt, ahol az ilyen újságok is készülnek: szerkesztőségekben és nyomdák előkészítő részlegeiben. A keresletnek megfelelően egyre nő a kínálat ezekből a rendszerekből és az esetek többségében az ajánlott rendszerek eléggé hasonló módon épülnek fel: egy, a célfeladathoz illő teljesítményű számítógépre installálják az adott témakörhöz tartozó néhány slágerszoftvert, csatlakoztatnak a rendszerhez különböző perifériákat, nyomtatót, szkennert, stb. — és már kész is a célmunkahely.



Az így felépített számítógépes munkaállomás az installálás és átadás ideje alatt rendezett képet mutat, de ha a rendszer installálója mondjuk néhány hónapos intenzív használat után belenéz a merevlemez alkönyvtáraiba, elsírja magát. A dolog ekkor valahogy úgy néz ki, hogy a merevlemez alkönyvtáraiban, beleértve a gyökérkönyvtárat is, rendszertelenül elszórva meg lehet találni az elmúlt időszak összes létrejött dokumentumait, anyagait: rendet tenni szinte lehetetlen a sok kereszthivatkozás miatt. Káosz!

Bár az entrópia így is, úgy is - mint tudjuk - növekszik, megfékezésére az egyik leghatékonyabb eszköz lehet a partners® Hungary Kft. saját fejlesztésű magyar szoftvere, keretrendszere: a HEADLINE.

A Headline szerzői többéves, szerkesztőségi rendszerek tervezésében és installálásában összegyűjtött tapasztalataik felhasználásával egy olyan programot hoztak létre, amely a felhasználók igényeit messzemenően figyelembevéve alkalmazható éppúgy orvosi rendelő vagy autókereskedés kartoték-rendező keretszoftvereként, mint újságszerkesztőség munka-szervező rendszereként. Felépíthető vele akár egygépes munkahely, akár sokszáz gépes nagy hálózati rendszer, mindkettőn áttekinthetően összefogja a napi felhasználás során keletkezett anyagokat és kényelmessé teszi ezek (újra) használatát.

OFF-LINE verzió

A Headline OFF-LINE verziója egygépes munkahelyek kiépítésére alkalmas, melyen a tervezett, megvalósítandó feladat minden részfeladata elvégezhető. Az OFF-LINE verzió segítségével kényelmesen végigkísérhetjük az egyes anyagok (például levelek, vagy bonyolultabb szöveges és képfileok, tördelt oldalak) útját keletkezésüktől elkészülésükig. Minden munkafázis egyetlen munkahelyen található meg. A különböző állapotban lévő anyagok mindig azonnal elérhetők. Minden anyagról plusz információ áll rendelkezésre az anyagok Információs lapján.

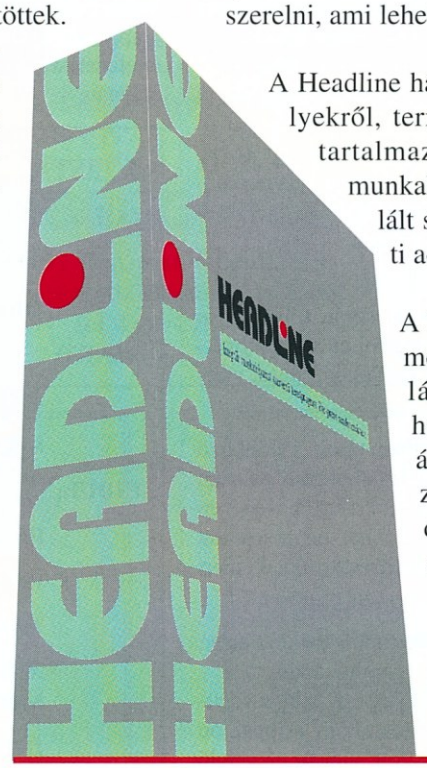
MIT TUD A HEADLINE?

- A Headline keretrendszer, sok kényelmi szolgáltatással.
- Minden, az adott munkahelyre installált szoftver futtatható Headline alól.
- A létrejött file-okat információs lappal együtt adatállományokba szervezi, áttekinthetően tárolja, ezáltal elősegíti intelligens visszakeresésüket és az elavult anyagok törlését.
- A Headline anyagorientált rendszer: a felhasználó a kívánt anyag kiválasztása után egy mozdulattal kezdeményezheti annak feldolgozását egy másik programmal, függetlenül attól, hogy az adott anyag melyik programmal lett létrehozva.
- Beépített automatizmusaival megkönnyíti a napi munkát.
- Könnyen kezelhető és tanulható (fool-proof).
- Mivel shell jellegű feladatokat is ellát, a felhasználónak nem szükséges az operációs rendszer működését ismernie.
- Egygépes munkahely alakítható ki az OFF-LINE verzió segítségével.
- Hálózati verziója NOVELL hálózatban futtatható, akár merevlemez nélküli hálózati munkahelyekről is.
- Az egész országot átfogó lokális hálózatok összekapcsolásával létrehozott rendszerben a különböző helyszínek közötti adatcserét és a helyi munkaszervezést egyszerre képes biztosítani.

KÉTFÉLE MÓDSZERE

2
Végy egy jó rendszer-programot

Az OFF-LINE verzió gondoskodik arról is, hogy a rendszer merevlemezén rend legyen, az elavult anyagokat floppy-ra menthessük, vagy, ha végleg nincs rájuk szükség, akkor törölhessük azokat.
A Headline OFF-LINE verziója ajánlott például kisebb irodákban (legyen az akár egy utazási iroda), ahol a rendezés vagy a dokumentumok, kiadványok előkészítése egy, esetleg két párhuzamosan futó munkahelyen megoldható. A Headline ajánlott olyan cégek számára is, akik törődnek az általuk kiadott anyagok küllemével és a számítógépes kiadványszerkesztés mellett döntöttek.
Ajánlott akár egy menedzser vagy vállalkozó számára is, aki elhatározta, hogy az üzletvitelével kapcsolatos számítógépes információkat (leveleket, faxokat, stb.) jól áttekinthető, könnyen hozzáférhető formában szeretné tárolni.



HÁLÓZATI verzió

A Headline hálózati verziója NOVELL hálózaton futtatható. Az egyes hálózati munkahelyeken ilyenkor mindenütt a program egy helyi verziója fut, és a hálózat központi gépén találhatók a központi adatállományok. A munkahelyeken lehetőség van mind a központi gépen található adatállományok, mind pedig a helyi gép saját merevlemezén elhelyezkedő helyi adatállományok elérésére. Az egyes munkahelyek egy-egy célfeladat elvégzésére specializálódhatnak és az anyagok feldolgozási folyamatának egy-egy állomását, lépcsőjét képviselhetik. Az optimális munkamenet ilyen esetben az, hogy a beviteli munkahelyeken megszületnek az anyagok és természetesen, a mindenki által elérhető központi adatállományokba kerülnek. Az egyes munkahelyek a feldolgozási folyamatnak megfelelően sorban elvégzik rajtuk a részfeladatokat, majd a feldolgozási folyamat utolsó állomására kerülve elkészíthető a végtermék. Ezek után a már kész anyagok egy ideig őrizhetők a rendszerben és ha végleg nincs rájuk szükség, akkor törölhetők. Az ilyen munkaszervezésben a Headline hálózati verziója szinte nélkülözhetetlen. Az egyes anyagokhoz az elkészültségi állapotuknak megfelelő, státusz jellegű információt rendel, így biztosítja azt, hogy a különböző munkahelyeken ülő felhasználók csak azokat az anyagokat érjék el, amelyekkel dolgozniuk kell. Megszervezi az anyagok áramlását a rendszerben és az elkészült anyagokat elkülöníti a feldolgozásra váróktól. Mivel a különböző munkafolyamatok különböző teljesítményűek, az egyes munkahelyeket az adott feladatnak megfelelő hardverrel lehet felszerelni, ami lehetőséget ad a költségek optimalizálására.

A Headline hálózati verziója olyan hálózati munkahelyekről, terminálokról is futtatható, amelyek nem tartalmaznak beépített merevlemezt. Az ilyen munkahelyek a hálózat központi gépén installált szoftvert futtatják és kizárólag a központi adatállományok elérésére alkalmasak.

A Headline hálózati verziója segítségével megoldható különböző városokban installált és egymással összekapcsolt helyi hálózatok között az anyagok ON-LINE áramoltatása is. A Headline hálózati verziója ajánlott komolyabb gépparkkal rendelkező vállalkozások számára, ahol már működik számítógépes hálózat, vagy érdemes egy NOVELL hálózatot telepíteni.

Ha ez a rövid leírás felkeltette az Ön figyelmét és további információkat, árakat szeretne megtudni a Headline-ról, esetleg működés közben szeretne megtekinteni HEADLINE-nal felépített rendszereket, kérjük, hívjon minket bizalommal.

partners® Hungary Kft.
1149 Budapest, Angol u. 6.
Tel.: 221-5123, 221-5126 Fax: 251-6127

partners® Hungary

HEADLINE: A RENDSZERE

Minden kedves érdeklődőt és viszonteladót szeretettel várunk a PrintExpo '94 kiállításon, október 11-től 15-ig a D pavilon 403/F standján.

ASSEMBLER ISKOLA

Mindazok számára, akik szeretnének megtanulni valamilyen programozási nyelvet, segítséget nyújt folyóiratunk. Elsőként inkább a gyakolati, mint elméleti oldaláról ismertetjük az assembly nyelvet.

Ez az a nyelv, ahol egy program elkészítéséhez nem elég csupán az utasítások ismerete, hanem egy átfogó hardware-ismeretre is szükség van. Az assembly a számítógép saját nyelvén való programozása.
Egy program megírása tulajdonképpen nem más, mint az adott feladat elemeire való bontása. Ezek az elemek magasabb szintű nyelveknél nagyobbak, az assemblynél a lehető legapróbbak. Ebből is látszik, hogy nem tartozik a könnyen megtanulható nyelvek közé. A programozás során több új fogalommal kell majd megismerkednünk, amik egyben a programozó eszköztárát is képezik. Ezek határozzák meg a számítógépes programozás jellegét.
A programozás eszközei:

Eljárások: olyan rutinok, amire a program során többször is szükség van. Ilyenkor azt elegendő egyszer megírni és a továbbiakban csak hivatkozni rá. Ilyen eset például egy szöveg kiíratása, vagy egy vonal rajzolása, stb.

Változók: nem mások, mint a memóriában tárolt különböző típusú adatok. Egyes feladatok megoldása elképzelhetetlen pusztán regiszterek használatával, a memória mérete pedig bőségesen megengedi, hogy abban adatokat tároljunk. Ezeket ne tévesszük össze a magasabb szintű nyelveknél megismert változókkal, mert nagyon sok különbség van köztük.

Ciklusok: ez egyike a leggyakrabban használt eszközöknek, ugyanis segítségével az egymás után többször végrehajtandó programrészeket elegendő csak egyszer megírni és egy ciklus segítségével többször végrehajtani. Például az a feladat, hogy csippanjon ötöt a gép, akkor nem kell ötször megírni a rutint, hanem elegendő egyszer és utasítani a gépet arra, hogy azt ötször hajtsa végre.

Feltételek vizsgálata, elágazások: igen gyakran előfordul, hogy el kell döntenünk, hogy egy eredménnyel mit kezdjen a gép, vagy adott helyzetben választani kell a különböző lehetőségek közül. Erre a célra szolgálnak a feltételek és elágazások.

Alapértelmezés szerint a számokat decimálisan írjuk. Ha ettől eltérően ábrázoljuk azokat, akkor jelölni kell, hogy az adott szám melyik számrendszer szerint értelmezendő, így a bináris számok után egy 'b', hexadecimális számok után pedig egy 'h' betűt. A 16 bites regisztereket használjuk címzésre, egy 16 bites számmal megcímezhető legnagyobb memóriacím a 65535, azaz 64 KByte. Nos ennél még a leggyengébb XT-ben is több van. A megoldás az, hogy a gépben lévő memóriát felszeleteljük és laponként kezeljük. Így jött létre a szegmentált címzés, amihez szükség van egy szegmens és egy index címre. Ennek lényege, hogy a szegmenscím segítségével kiválasztunk egy 64K-s lapot a memóriából, és az indexcím segítségével határozzuk meg a pontos címet. A teljes memóriacím 20 bit hosszú. Ez úgy alakul ki, hogy a szegmenscímhez képest négy bittel el van tolva az indexcím. Ebből adódóan két szomszédos szegmens egymástól 16 Byte (egy paragrafus) távolságra van. A megoldás hátránya, hogy csak un. paragrafushatáron kezdődhet egy szegmens. A tanuláshoz szükségünk lesz egy assembler-fordítóra (a Turbo Assemblert ajánljuk), valamint egy egyszerű szövegszerkesztőre. Először minden magyarázat nélkül közlünk egy programlistát az 1. listán és a fordításhoz szükséges parancsfilet a 2. listán.

1. LISTA.

```
CODE    SEGMENT PUBLIC 'CODE'
        ASSUME CS:CODE , DS:CODE
        ORG    100h                ; COM file-t készítünk.
START:
        JMP    INDUL
        SZOVEG DB    ' Helló világ ! $'
INDUL:
        PUSH   CS
        POP    DS
        MOV    AH,09h              ; DOS funkció beállítása
                                   ; 09h = String kiíratása.
        MOV    DX,offset SZOVEG; Kiírandó string kezdete
        INT    21h                 ; DOS megszakítás
                                   ; hívása
        MOV    AX,4C00h            ; 4Ch = Kilépés
        INT    21h                 ; DOS megszakítás
                                   ; hívása
CODE    ENDS
END     START
```

2. LISTA

Turbo Assembler-hez COM-ra fordító BAT-file.

```
TASM %1
TLINK -T %1
DEL %1.OBJ
DEL %1.MAP
```

5 hónap alatt megtanítjuk játékprogramot írni !!!

Azt hiszem egy programozó számára a legszebb megmérettetés egy működő játékprogram elkészítése. Ennek a sorozatnak az a célja, hogy segítséget nyújtson ehez a munkához. Közzétesszük a teljes forrásanyagot, és a működési elvnek a leírását, így mindenki saját elképzelései szerint alakíthatja át, vagy fejlesztheti. A közölt forráslisták assemblerben lesznek, de ha igény lesz rá, akkor C és Pascal változat is elkészíthető.

Szükséges hardware feltételek:

Úgy gondolom ma már eléggé elterjedt, és mindenki számára hozzáférhető konfigurációról van szó. A minimum feltételek:
- AT-286 alaplap
- VGA videókártya
- 300 Kb üres lemezkapacitás

Játék ismertetése:

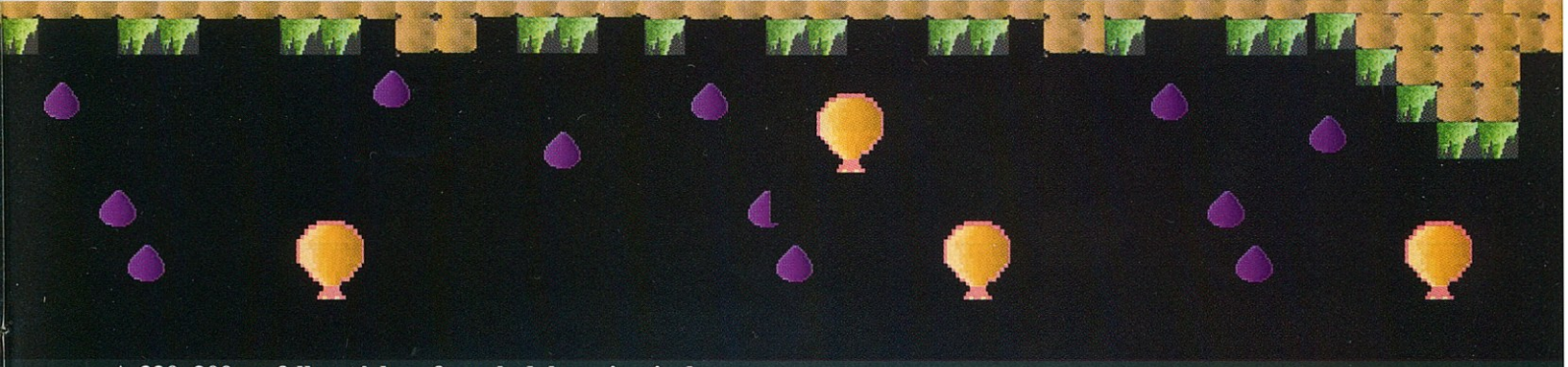
A játék tulajdonképpen a síkban játszódik, kétdimenziós téglákkal és figurákkal. A teljes játéktér 16 darab szobára van felosztva -- ami tetszőlegesen módosítható --, és ezeken keresztül kell haladni, hogy teljesíteni tudjuk a játékot. Találkozunk ellenséges figurákkal, akikhez nem szabad hozzáérni, mert különben fogy az energiánk. Megismerkedhetünk a billentyűzet kezelésével, a videógrafika kisebb trükkjeivel. Az itt felhasznált spriteok például bármely más programban felhasználhatók, mivel méretük horizontálisan és vertikálisan is módosítható. Megoldható prioritásuk és képük. Szó lesz a file-kezelésről és a hangszóró programozásáról is. Vagyis áttekinthető és működő példákat találhatunk, amiket más területeken hasznosíthatunk.

A program elkészítése:

A program futása közben teljesen át kell venni a vezérlést, mivel az NMI-t is le fogjuk tiltani. Erre azért van szükség, nehogy egy rezidens program ellopja az időnket futás közben, és ez az a módszer amivel teljesen átvehetjük az ellenőrzést a perifériák felett. Most, az első részben azokat a rutinokat szeretném bemutatni, amelyek még az inicializálás során hajtódnak végre. Az inicializálás az amikor a program elején különböző dolgokat beállítunk. Ilyenek saját rendszerváltozóink, a videóüzemmód és néhány interrupt vektor. Ezeket természetesen külön-külön is felhasználhatjuk más programokban, mivel próbáltam úgy megírni őket, hogy kiragadva környezetükből más programokban is működjenek. Ez persze nem mindig teszi ideálissá a szubrutint, de a működési feltételeknek eleget tesznek. Tehát kezdjük egy pár valószínűleg egyszerű, de kötelező jellegű procedúrával. Ha ezekkel megismerkedtünk, akkor jöhet majd a keretrendszer amibe ezeket beleillesztjük.

A videómód beállítása:

A videómód beállításához túl sok ismeret nem szükséges.



A 320x200-as felbontásban fogunk dolgozni, mivel ezt az üzemmódot minden VGA kártya ismeri. A képernyő kezelésénél soha ne feledkezzünk meg a felbontásról, ugyanis a képkialakításnál elengethetetlenül fontos lesz. Ahhoz, hogy a program befejezése után zökkenőmentesen térjünk vissza a DOS-hoz, az is fontos, hogy ne felejtsük el visszaállítani a indítás elötti videóüzemmódot. Így lesz két egyszerű rutinunk amik beállítják ezeket az üzemmódokat. Mivel az üzemmód elmentéséhez elegendő egy byte, így elég ha egy egybyte-os memóriaváltozót hozunk létre. Ide el tudjuk menteni a videómódot a program elején, a végén pedig vissza tudjuk állítani azt amiből indultunk. lásd 1. lista.

Képernyő törlése, színek:

Elössző is egy pár elmaradhatatlan fontos dolog. A videómemória az A000h segmens-címen és a 0000h ofszetcímen kezdődik. A programban a videómemoria megcímzésére általában az ES és valamely egyéb pointer-regisztert fogjuk használni.

Tehát a segmens címet az ES-be, a relatív offsetcímet meg például a DI vagy az SI regiszterbe tesszük. Mivel a videómemória fizikai felépítése igen egyszerű, így nincs túl sok dolgunk. Egyszerűen szervezünk egy ciklust 0-tól 32000-ig, mivel ennyi darab word van a "képernyőn". Egy byte egy pontként jelenik meg, így ha mindegyikbe nullát töltünk akkor az egész képernyőt letöröltük feketére. 2. lista.

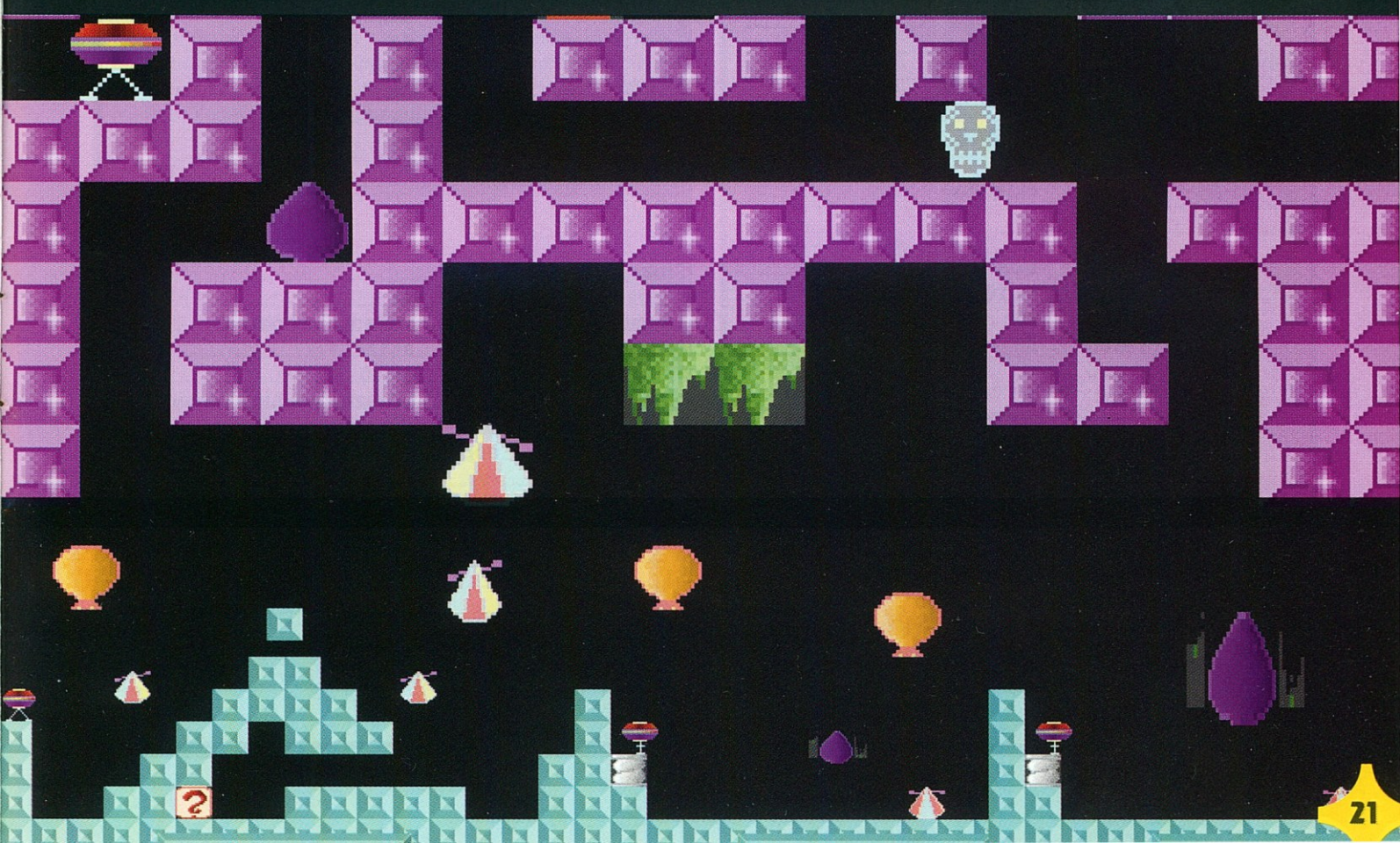
Hogy mért pont feketére színezi ha nullát töltünk bele? Mivel egy byte ba 0-255-ig tudunk számot tölteni, ezért van pontosan 256 darab színünk. Minden egyes szám megfelel egy RGB színnek, Érdemes kipróbálgatni. Paletta beállítása a 3. listán.

Egy-két szubrutin:

Szükségünk lesz néhány apró rutinra. Ilyen például az INKEY nevű, amely nem csinál mást mint azt, hogy ha átadjuk neki a vezérlést, akkor addig vár míg meg nem nyomunk egy billentyűt. Erre sokszor szükség lehet. Tulajdonképpen azt vizsgálja, hogy felengedtünk-e egy billentyűt, mert ez egyszerübb. Ez azért van mert a billentyűzetet kezelő 8255-ös PPI minden egyes lenyomáskor és felengedéskor generál interruptot. Lenyomásnál ilyenkor megjelenik a 60h porton a billentyű kódja, felengedéskor a kód+80h. Ebből az következik, hogy az alsó 7 biten van ábrázolva a billentyűkód, és a 8. bit állapota jelzi, hogy azt éppen lenyomták vagy felengedték., erre a billentyűkezelésnél visszatérek.

A WAITS rutin csak egy egyszerű várakozás, aminek ideje az IDO nevű változótól függ. 4. lista.

Szalay Zsolt



1. LISTA

```
VidIni    Proc
    xor         al,al
    mov         ah,0fh
    int         10h             ;videó üzemmód
                                ;lekérdezése
    mov         vidmod,al       ;üzemmód elmentése
    mov         ax,0013h
    int         10h             ;320x200 256 szín
    ret
    EndP

VidUnIn Proc
        mov             al,vidmod
        xor             ah,ah
        int             10h
        ret
        EndP
```

2. LISTA

```
ClsProc
        push    cx
        push    ax
        mov     cx,32000        ;320*200 db byte =
                                ;32000 word a
                                ;képernyõ memória
                                ;hossza
        xor     ax,ax
        xor     di,di
cls1:   mov     es:[di],ax
        add     di,2    ;word-önkénti léptetés
        loop    cls1    ;32000-szer visszaugrik
        pop     ax
        pop     cx
        ret
        EndP

Cls     Proc
        push    cx
        push    ax
        mov     cx,32000                xor
ax,ax
```

3. LISTA

```
RGBbe  Proc
                mov     bx,0
                mov     si,0
                mov     di,offset rgb
                ;az RGB tábla offsetcíme
                ;segmenscíme a DS-ben
                ;Felépítése: 256 db
                ;Red,Green,Blue 1-1-1 byte
                ;értékük 0-63-ig
rgbc1:  push    si
                mov     dh,[di+bx]
                inc     bx
                mov     ch,[di+bx]
                inc     bx
                mov     cl,[di+bx]
                inc     bx
                push    bx
                mov     bx,si
                mov     ax,1010h
                int     10h     ;RGB beállítása
                pop     bx
                pop     si
                inc     si
                cmp     si,256
                jne     rgbc1   ;rgb beállítása
                ret
                EndP
RGBbe  Proc
                mov     bx,0
                mov     si,0
                mov     di,offset rgb
                        ;az RGB tábla offsetcíme
                        ;segmenscíme a DS-ben
```

4. LISTA

```
WaitS   Proc
                push    dx
                mov     dx,ido
wa1:            dec     dx
                cmp     dx,0
                jnz     wa1
        ;amíg dx nem nulla vissza
                pop     dx
                ret
                EndP


Inkey           Proc
                push    ax
inke0:  in      al,60h  ;billentzüzet portja
                and     al,128
                jnz     inke0
                        ;visszaugrik ha felengedtek
                        ;egy billentyût
                pop     ax
                ret
                EndP
```

UNIX Rovat

Ebben a sorozatban a UNIX shell használóinak, programozóinak szeretnénk fórumot biztosítani. Lehetővé szeretnénk tenni gyakorlattal rendelkezőknek a tapasztalatcserét és a kezdők találkozását életszagú mintákkal. Szívesen veszünk minden platformról forrásokat, észrevételeket, tapasztalatokat és kéréseket.

Aki már használta a UNIX shellt, annak nem kell bizonyítani, hogy a mai formájában rendelkezik ugyanazokkal a képességekkel, mint a 4. generációs programnyelvek. Könnyen tanulható. Rendkívül tömör programokat készíthetünk. Fájlok csoportjait kezelhetjük egyszerre. A bővítése egyszerű: saját shell kisügyeseink mellett, akármely a gépünkön rendelkezésre álló programnyelven megírhatjuk kiegészítéseinket. Az awk, grep, sed segítségével meglepően bonyolult adatkezeléseket is meg lehet oldani. A fejlesztésre fordítandó idő lényegesen kevesebb, mint bármely más programnyelv esetén. Prototípusok, minták ismeretében még hatékonyabbá tehető a fejlesztői munka. Annak a számára, aki ezt megtanulta, járatos benne, minden UNIX rendszer nyitott, szabad a mozgástere.

Persze nem futási sebességrekordokat kívánunk megdönteni, de a gyors eredmény mindenkit kárpótol. Ne ítéljünk előre. A futási sebességgel kapcsolatban kellemes meglepetések érik a rutinos shell programozókat.

A bonyolultabb rendszerek telepítése is shell programok segítségével történik. Ezek akár néhány száz k méretűek is lehetnek. ld Lotus 123, Wordperfect, Dataflex Ne ijedjünk meg tőlük, kíváló ötlet tárházak!

Tanuláshoz, hibakereséshez használjuk az operációs rendszerrel kapott kézikönyvet, vagy amennyiben érdeklődés mutatkozik, néhány szakkönyvet a későbbiekben ajánlhatunk.

Indítsuk ezt a sorozatot egy egyszerű programmmal, amely a magyar ékezetes karakterek használatát segíti. Hasznos minden olyan eszközök használatában, amely az ASCII tábla 127 felett bizonyos kódokat saját maga értelmez. pl. nroff,troff

```
1:#!/usr/bin/ksh
2:# Készült COHERENT 4.0-ra
3:# Magyar karakterek át és vissza-
csoportosítása
4:# 1993.aug."BECO"Kft.
5:while test $# -gt 0
6:do
7:     case $1 in
8:          -mc) arg=$1;;
9:          -ms) arg=$1;;
10:         -n*) o1=$1;;
11:         -r*) reg=$1;;
12:         *) szveg=$1;;
13:    esac
14:shift
15:done
16:
17:# az óúíş marad, a többi átalakul:
18:
19:cat $szveg\
20:|tr  "öüôűéáÖÜň˙ůÉĹěä" "Ľ« ń˝»®Ń¬żŞ˘ "\
21:|nroff $arg $o1 $reg\
22:|tr "Ľ«_ń__˝»®Ń¬żŞ˘_" "öüôűéáÖÜň˙ůÉĹěä"
```

Második programunk egy ügyes kurzor pozicionáló módszert mutat be.

a parancssor legyen:

```
$cursor sor oszlop
```

A kurzor pozicionálás terminál függô, a módját megtaláljuk a `/etc/termcap` fájlban, (`cm`) bejegyzés alatt. Hozzunk létre egy CURSOR nevű környezeti változót, `cursor` ezt fogja megtalálni környezetében. Egy pc terminál részére írjuk be az alábbi sort a .profile fájlunkba. (`^[` az `ESC` karakter, vagy `Ctrl ]`)

```
CURSOR=^[[%d\;%dH
```

A `cursor` program:

```
1:#!/usr/bin/sh
2:# készült COHERENT 3.2-re
3:# 1992.maj."BECO"Kft.
4:# !/usr/bin/ksh, COHERENT 4.0-ra OK!
5:# 1993.aug."BECO"Kft.
6:USAGE='usage: cursor sor oszlop'
7:EUSAGE=1
8:
9:        if test "$#" -ne 2
10:       then
11:       echo $usage 1>&2
12:       exit $EUSAGE
13:       fi
14:
15:echo `echo "$CURSOR"|
16:sed    "s/%./$1/
17:       s/%./$2/"`\\c
```

A `cursor`, `clear`, `echo` segítségvel néhány sorban látványos dolgokat rajzolhatunk a képernyőre, és a `read` segítségével beolvashatjuk a felhasználó válaszait. A képernyőn mezőket törölhetünk a kurzor mozgatásával és betűközök írásával. Ezzel a módszerrel a képernyő szinezése is megvalósítható. Ötleteket várunk.

Legközelebbi alkalommal a `cursor`-nak egy továbbfejlesztett, gyorsabb változtát fogjuk bemutatni.

Várjuk leveleiket, ötleteiket. Ne feledjék el megadni, hogy programjuk eredeti változata mely operációs rendszer alatt készült és lett kipróbálva.

Berta Sándor
tel:217-4578
"BECO"Kft.

Forráskód

Tel.: 134-1273

Program Börze

C++

View

Turbo C++ for Windows Visualized 9.800.-

Borland C++ for OS/2 49.800.-

Fortran

MS Fortran 5.1
16.400.

C++

View

Borland C++ 4.0 CD 19.900.-

Borland C++ 4.0 3,5 28.800.-

Visual C++ 1.5 prof CD 48.000.-

Assembler

Windows App

Turbo Assembler 4.0
Dos / Win 11.800.-

Visual Basic

Visual Basic for Windows 3.0 standard 16.800.-

Visual Basic for Windows 3.0 professional CD 42.000.-

Network

Pascal

Set Up...

Turbo Pascal 7.0
16.800.-

Turbo Pascal for Windows 1.5
16.800.-

Borland Pascal with Object 7.0 Dos / Windows
34.000.-

Help

Adatbázis kezelők

dBase 5.0 for Dos 36.000.-

dBase 5.0 for Windows 36.000.-

Paradox 4.5 for Windows 18.000.-

Change All

Foxpro 2.6 for Dos standard 9.900.-

Add

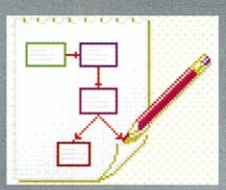
Foxpro 2.6 for Windows standard 9.900.-

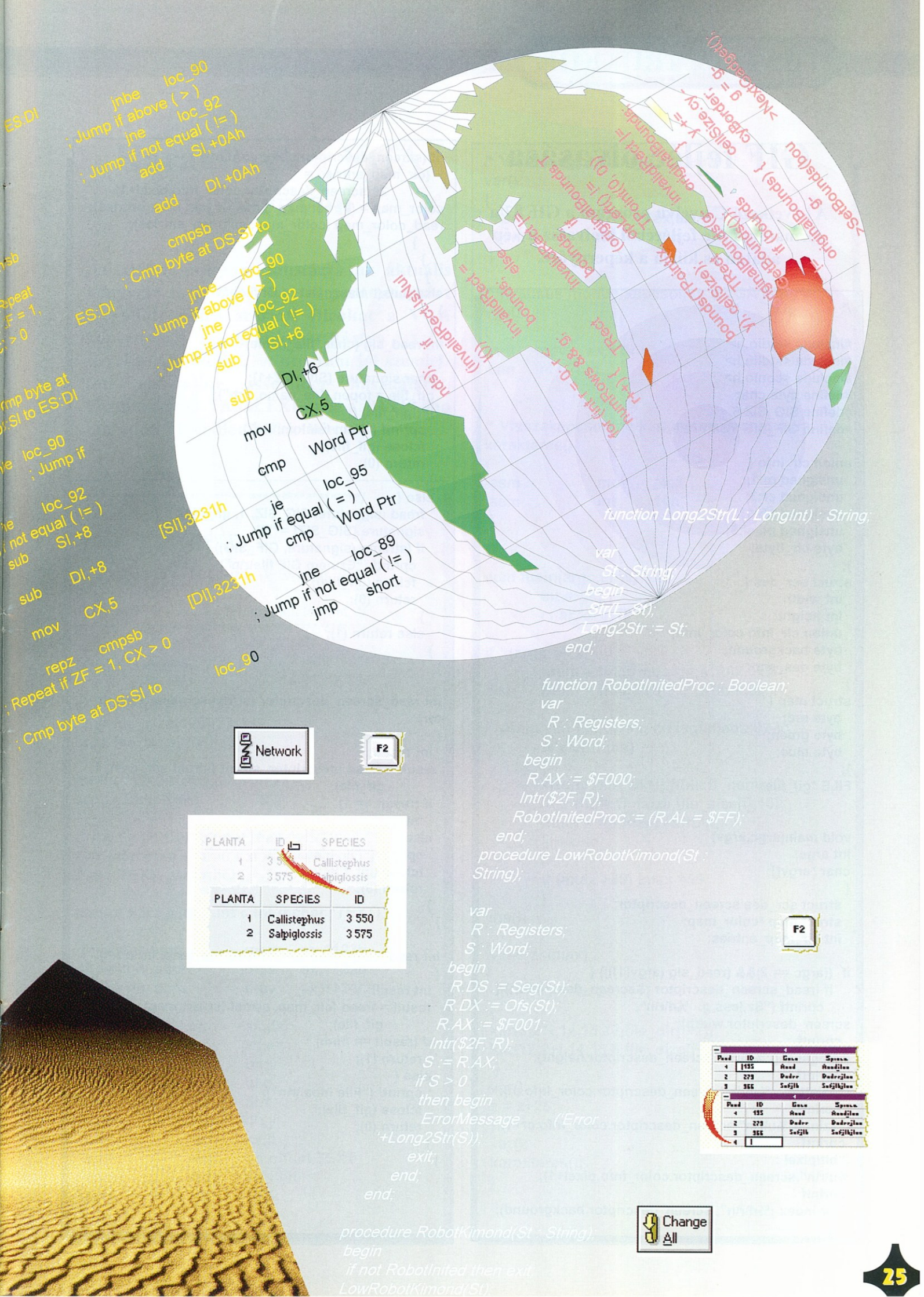
Foxpro 2.6 for Windows professional 59.900.-

Assembler

Windows App

Macro Assembler 6.1 22.800.-




jnbe loc_90
; Jump if above (>)
jne loc_92
; Jump if not equal (!=)
add SI,+0Ah
add DI,+0Ah
cmpsb
; Cmp byte at DS:SI to
ES:DI
jnbe loc_90
; Jump if above (>)
jne loc_92
; Jump if not equal (!=)
SI,+6
sub
DI,+6
sub
CX,5
mov
Word Ptr
cmp
loc_95
je
; Jump if equal (=)
Word Ptr
cmp
loc_89
jne
; Jump if not equal (!=)
short
jmp
[SI],3231h
[DI],3231h
loc_90
; Repeat if ZF = 1, CX > 0
repz cmpsb
; Cmp byte at DS:SI to
function Long2Str(L : LongInt) : String;
var
St : String;
begin
Str(L, St);
Long2Str := St;
end;
function RobotInitedProc : Boolean;
var
R : Registers;
S : Word;
begin
R.AX := $F000;
Intr($2F, R);
RobotInitedProc := (R.AL = $FF);
end;
procedure LowRobotKimond(St :
String);
var
R : Registers;
S : Word;
begin
R.DS := Seg(St);
R.DX := Ofs(St);
R.AX := $F001;
Intr($2F, R);
S := R.AX;
if S > 0
then begin
ErrorMessage ('Error
'+Long2Str(S));
exit;
end;
end;
procedure RobotKimond(St : String);
begin
if not RobotInited then exit;
LowRobotKimond(St);
Network
F2
PLANTA ID SPECIES
1 3 550 Callistephus
2 3 575 Salpiglossis
PLANTA SPECIES ID
1 Callistephus 3 550
2 Salpiglossis 3 575
F2
Change All

GIF fejléc olvasása

A következő C nyelvű program a GIF 87/a formátumú file fejlécéből kiolvassa a kép adatait és kiírja a képernyőre.

```
/*        gi.c      */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define byte char
#define SIG_SIZ 6
#define GIF_SIG "GIF87a"

union clr_info {
  unsigned m:1;
  unsigned cr:3;
  unsigned ub:1;
  unsigned pixel:3;
  byte clr_byte;
};
struct scr_des {
  int width;
  int height;
  union clr_info color_info;
  byte background;
  byte des_end;
};
struct map {
  byte red;
  byte green;
  byte blue;
};
FILE *gif_file;


void main(argc,argv)
int argc;
char *argv[];
{
  struct scr_des screen_descriptor;
  struct map *color_map;
  int clr_map_entries;

if  ((argc == 2)&& (read_sig (argv[1]))) {
    if (read_screen_descriptor (&screen_descriptor)) {
      cprintf ("Sz,less,g:  %d\r\n",
screen_descriptor.width);
  cprintf
("Magasság:%d\r\n",screen_descriptor.height);
  cprintf
("paletta:    %u\r\n", creen_descriptor.color_info.m);
  cprintf
("szinbit:%u\r\n",screen_descriptor.color_info.cr+1);
 cprintf
("bit/pixel :
%u\r\n",screen_descriptor.color_info.pixel+1);
 cprintf
("clr index :%d\r\n", screen_descriptor.background);
 if (screen_descriptor.color_info.m)
  {clr_map_entries =
          2^(screen_descriptor.color_info.pixel+1);
color_map = (struct map *) malloc (clr_map_entries);
read_color_map (color_map, clr_map_entries);
     }
   }
  }
else puts(" Használat :   GI filenév ");
}

int read_sig (char *file_name)
{
  char signature [SIG_SIZ+1];
  gif_file = fopen (file_name, "rb");
  if (!gif_file) {
    cprintf ("Nem találom.\r\n");
    fclose (gif_file);
    return (0);
  }
  else {
    fread (signature, SIG_SIZ, 1, gif_file);
    signature [SIG_SIZ] = '\0';
    if (strcmp (signature, GIF_SIG)) {
      cprintf ("Ez nem GIF file\r\n");
      fclose (gif_file);
      return (0);
    }
    else return (1);
  }
}

int read_screen_descriptor (struct scr_des *descrip-
tor)
{
  int result;
 result = fread (descriptor, sizeof (struct scr_des), 1,
          gif_file);
  if (result == 1)
    return (1);
  else {
    cprintf ("File hiba.\r\n");
    fclose (gif_file);
    return (0);
  }
}

int read_color_map (struct map *clr_map, int num)
{
  int result;
  result = fread (clr_map, sizeof (struct map), num,
          gif_file);
  if (result == num)
    return (1);
  else {
    cprintf ("File hiba.\r\n");
    fclose (gif_file);
    return (0);
  }
}
,
```

Karakterek a VGA képernyőn

Az alábbi program kilistázza az aktuális könyvtár tartalmát, vagy a paraméterként megadott szűrő szerinti fileokat a VGA képernyőn a grafikus kártya saját karakterkészletével. A program jól szemlélteti az inline assembler használatát. Fordításkor a TASM.EXE-nek a PATH-ban megadott útvonalon kell lennie.

```
/***********************************************/
/*                                             */
/*                vgadir.c                     */
/*                                             */
/*    fordítás :  TCC -mt -lt   vgadir         */
/*                                             */
/* TASM.EXE -nek  PATH ban kell lennie!  */
/***********************************************/

#include <dir.h>

char file_dir[100][13];
char file_size[100][8];
char file_date[100][13];

int fnum,i=0,z,db,j,c,h,v,imax=0;
struct ffblk ffblk;

/* Az x,y pontba p pont magas betűvel */
/* c színben d számú karaktert kiírunk */
/* a    TEXT  stringből                */

kiir(int x,int y,int p,int c,int d,char *TEXT)
{
 if(p==8)  asm          mov    ax,1123h
 if(p==14) asm          mov    ax,1122h
 if(p==16) asm          mov    ax,1124h
  asm            int      10h
  asm            mov      ax,ds
  asm            mov      es,ax
  asm            mov      bh,0
  asm            mov      bl,c
  asm            mov      dl,x
  asm            mov      dh,y
  asm            mov      cx,d
  asm            mov      bp,offset TEXT
  asm            mov      ax,1300h
  asm            int      10h
 }
```

```
/* Várunk egy billentyű leütésre  */
var()
{
  asm            xor      ax,ax
  asm            int      16h
 }

/* Grafikus módba kapcsoljuk a VGA kártyát  */
vga()
{
asm    mov    ax,12h
asm    int    10h
}

/* Visszakapcsolunk karakteres üzemmódra */
karakteres()
{
 asm            mov      ax,3
 asm            int      10h
 }


void main(argc,argv)
          int argc;
          char *argv[];
{
if   (argc == 2)
          fnum = findfirst(argv,&ffblk,0x3f);
else      fnum = findfirst("*.*",&ffblk,0x3f);

/* Amíg van file olvas, de legfeljebb 99-ig */
          while(!fnum)
          {
          strcpy (file_dir[i],ffblk.ff_name);
          ltoa (ffblk.ff_fsize,file_size[i],10);
          fnum = findnext (&ffblk);
          imax=i;
          i++;
          if (imax >99) imax =99;
          }
vga();

while(z<imax)
{
   for (i=0; i <= 24; i++)
 {
 kiir(3,2+i,14,15,12,file_dir[i+z]);
 kiir(17,2+i,14,15,6,file_size[i+z]);
  }
 var();
 z = z+25;
 }
 var();
 karakteres();
 }
```

File Edit View Table Properties Window Help

Egy legenda visszatér

Egyre másra jelennek meg a legendás szoftverek újabb verziói, és hála a Windowsnak egyre jobban hasonlítanak egymásra. Az új változatok egyre jobban kiszolgálják a felhasználót, mi marad akkor a gyalogos programozóknak?

Régi szép idők... Megjelenésekor Pc-s körökben mindenkit levett a lábáról a dBase 3+ fantasztikus teljesítményével, kezelési komfortjával.
Talán ennek köszönhető, hogy adatformátuma szabvánnyá vált. A következő IV. verzió nem sikerült igazán, közben jöttek a konkurensek egyre gyorsabb és jobb rendszerei. A fejlesztőknek előnyösebbek voltak azok a rendszerek amelyek önmagukban futtatható programokat produkáltak.
A dBase ennek ellenére a felszinen maradt, és most itt az új 5.0 változat ráadásul Windows alatt. A program remek, szinte mindent megtehetünk anélkül hogy egyetlen programsort kellene írni. A kompatibilítás egyirányú az új rendszer írja olvassa a régi fileokat. Az új DBF file fejléce megváltozott, a régi rendszerek nem ismerik fel.
Az új dBase igazi Windows-os program, annak minden előnyével. Kezelési komfortja, megjelenése az új Borland stílust idézi, szép ablakok még szebb kezelőszervek és mindez elegánsan könnyedén. A programozóknak jó hír, a rendszer köszönhetően a Windows-nak, tetszőleges programnyelven írt külső programokkal, dinamikus könyvtárakkal bővíthető. Természetesen megmaradt a saját nyelvű programozás lehetősége, a nyelv köszönhetően az új környezetnek, új lehetőségeknek jelentősen átalakult. Az adatkezelő és matematikai funkciókon kívül, megtalálhatók benne a Windows kezeléséhez szükséges függvények, szinte bármilyen alkalmazást fejleszthetünk vele Windows-os környezetbe. A kifinomult környezet és a dBase nyelv elterjedtsége miatt a program jó eséllyel indul a Foxpróval megvívandó harcban.

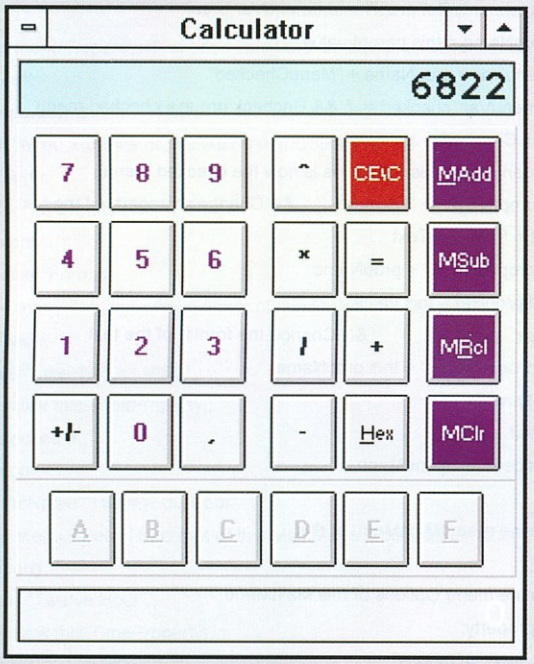
A forrásnyelven írt programot a dBase egy átmeneti bináris kódra fordítja, ezt a kódot futtatja akár egyszerre több példányban is. A Windows a futó alkalmazást önálló programnak látja, azonban ahogy lezárjuk a dBaset az alkalmazás is bezáródik.
A legnagyobb újdonságot magam részéről a képek, hangok, és általában tetszőleges Windows objektumok táblázatba illesztésének lehetőségét tartom. Természetesen ez azzal a hátránnyal járt, hogy meg kellett változtatni az adatfile formátumát. Erről a közeljövőben még ejtünk szót. Az adatfile létrehozása gyerekjáték az itt látható dialógus doboz segítségével

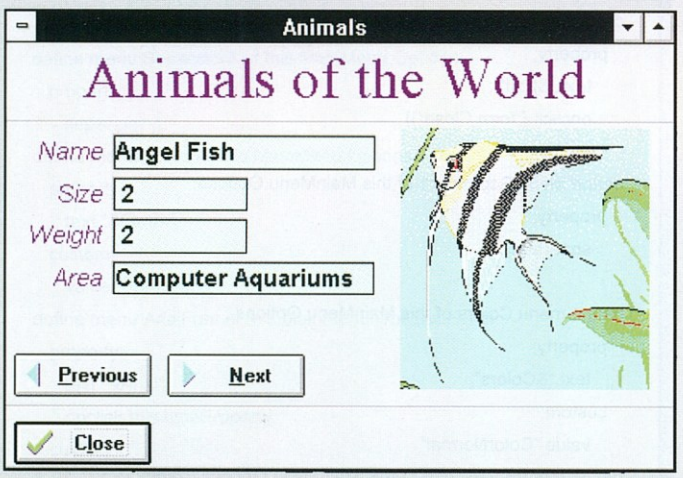
Field	Name	Type	Width	Decimal	Index
1	MEGNEV	Logical			None

Character
Numeric
Memo
Logical
Date
Float
OLE
Binary

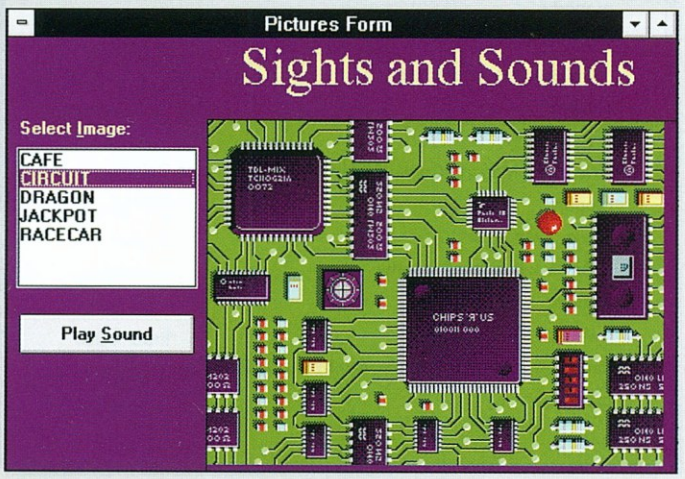
Az ábrán jól láthatók az új mezőtipusok.
A programmal való ismerkedést a kézikönyvön kívül a programba épített tanítómester is segíti, bár kissé erőszakos, néha nem hagyja magát lerázni.
Saját fejlesztésű programjainkat a rendszer önálló négy ablakos Debuggerével vallathatjuk, melyben az összes megszokott szolgáltatás elérhető, de egy kényelmes látványos környezetben.
A rendszernek természetesen beépített editora is van, a Borland C fordítóhoz adott példaprogramra emlékeztet. Összefoglalva: egy szép és szépreményű program az új dBase, rajtunk múlik mi lesz belőle.

Az új FORM GENERÁTOR tetszőleges tulajdonságú élő jelentéseket készít, akár kalkulátort is.





A táblázatokba az eddigi adatok mellett képeket, hangokat is beilleszthetünk természetesen ezek is megjeleníthetők a jelentésekben. Ezzel akár multimédia alkalmazásokat is készíthetünk dBase alatt.



Természetesen a programnyelv is bővült, változott. Az alábbi példaprogram a pontos időt jelzi ki egy ablakban.


```
*************************************************************************
* PROGRAM:     DBClock.prg
* WRITTEN BY:  Borland Samples Group
* DATE:        12/93    * UPDATED:     4/93
*************************************************************************
*
create session
set talk off
set ldCheck off
set design off
set procedure to program(1) additive
local f
f = new ClockForm()
f.open()

class ClockForm of Form
  this.top = 0
  this.left = 65.38
  this.height = 4.08
  this.width = 11.22
  this.mdi = .f.
  this.minimize = .t.
  this.text = "Clock"
  this.sizeable = .t.
  this.moveable = .t.
  this.sysMenu = .t.
  this.colorNormal = "bg+/n"

  define text timeText of this;
    property;
      text time(),;
      alignment 4,;
      colornormal "bg+/n",;
      height 6.12,;
      width 11.22,;
      fontName "Sans Serif",;
      fontBold .f.

  load dll dbtimer.vbx
  define dbtimer timer of this;
    property;
      Width      4.62,;
      Height     2.37,;
      Top      300.06,;
      Left     300.12,;
```

```
      OnTimer {;form.timeText.text = form.GetTime()}
  this.TimeProperty = CLASS::TimeProperty
  this.DefineMenu()

  procedure OnOpen

  private optionsRef

Public
formatMenuChecked,colorNormalMenuChecked,fontNameMenuChecked
,;
    fontBoldMenuChecked,fontItalicMenuChecked
  this.menuReleased = .f.          && -- Indicates that form has no menu
 this.timeFormat = 0              && -- Format of the time text
  this.GetTime = Get24Time          && -- Function called to display the
time.
                          && GetTime will be set to Get24Time or
                          && Get12Time depending on the format
                          && menu option selected.

 optionsRef = this.MainMenu.Options  && So don't have to access refer-
ence
                          && each time

  formatMenuChecked     = optionsRef.Formats.TwentyFourHrFormat
  colorNormalMenuChecked = optionsRef.Colors.CyanColor
  fontNameMenuChecked    = optionsRef.Fonts.SansSerifFont
  fontBoldMenuChecked    = optionsRef.Bold.BoldOff
  fontItalicMenuChecked  = optionsRef.Italic.ItalicOff
  this.OnSize(.f.,this.width,this.height)  && initial sizing
  this.timer.enabled     = .t.

  procedure OnClose
  set design on
rease
formatMenuChecked,colorNormalMenuChecked,fontNameMenuChecked
,;
    fontBoldMenuChecked,fontItalicMenuChecked
  close procedure program(1)
  procedure OnGotFocus
  set design off
  procedure OnLostFocus
 set design on
 procedure OnSize (nType, width, height)
 local t

  t = form.timeText
  t.width = width
  t.height = height
  t.fontSize = (height + width)/2

  procedure TwelveHour
  if this.timeFormat <> 12
    this.timeFormat = 12
    this.GetTime = Get12Time
  endif

  procedure TwentyFourHour

  if this.timeFormat <> 24
    this.timeFormat = 24
  this.GetTime = Get24Time
endif

procedure TimeProperty
private propRef,propName,t,menuVar,proc
propName = this.parent.value
menuVar = propName + "MenuChecked"
&menuVar..checked = .f. && Uncheck group's checked menu
this.Checked = .t.
&menuVar = this    && This is now the checked menu.
if propName <> "Format"        && Change a property of the text control
  t = form.timeText
  propRef = "t." + propName
  &propRef = this.value
else                          && Change the format of the text
  proc = "form." + this.procName
  &proc()
endif
procedure DefineMenu

define menu MainMenu of this

define menu Options of this.MainMenu;
  property;
    text "&Options"

define menu ExitMenu of this.MainMenu.Options;
  property;
    text "&Exit",;
    onclick {;form.Close()}

define menu Separator1 of this.MainMenu.Options;
  property;
    separator .t.

define menu Colors of this.MainMenu.Options ;
  property;
    text "&Colors";
  custom;
    value "ColorNormal"
define menu CyanColor of this.MainMenu.Options.Colors;
  property;
    text "&Cyan",;
    onclick this.TimeProperty,;
    checked .t.;
  custom;
    value "bg+/n"
define menu YellowColor of this.MainMenu.Options.Colors;
  property;
    text "&Yellow",;
    OnClick this.timeProperty;
  custom;
    value "gr+/n"
define menu MagentaColor of this.MainMenu.Options.Colors;
  property;
    text "&Magenta",;
    OnClick this.timeProperty;
  custom;
    value "rb+/n"
define menu RedColor of this.MainMenu.Options.Colors;
  property;
```

```
    text "&Red",;
    onclick this.timeProperty;
  custom;
    value "r+/n"

define menu Separator2 of this.MainMenu.Options;
  property;
    separator .t.
define menu Formats of this.MainMenu.Options;
  property;
    text "&Formats";
  custom;
    value "Format"
define menu TwentyFourHrFormat of this.MainMenu.Options.Formats;
  property;
    text "Twenty Four Hour",;
    onclick this.TimeProperty,;
    checked .t.;
  custom;
    procName "TwentyFourHour"
define menu TwelveHrFormat of this.MainMenu.Options.Formats;
  property;
    text "Twelve Hour",;
    onclick this.TimeProperty;
  custom;
    procName "TwelveHour"

define menu Separator3 of this.MainMenu.Options;
  property;
    separator .t.
define menu Fonts of this.MainMenu.Options;
  property;
    text "&Fonts";
  custom;
    value "FontName"
define menu ArialFont of this.MainMenu.Options.Fonts;
  property;
    text "&Arial",;
    onclick this.timeProperty;
  custom;
    value "Arial"
define menu ScriptFont of this.MainMenu.Options.Fonts;
  property;
    text "&Script",;
    onclick this.timeProperty;
  custom;
    value "Script"
define menu RomanFont of this.MainMenu.Options.Fonts;
  property;
    text "&Roman",;
    onclick this.timeProperty;
  custom;
    value "Roman"
define menu SerifFont of this.MainMenu.Options.Fonts;
  property;
    text "S&erif",;
    onclick this.timeProperty;
  custom;
    value "Serif"
define menu SansSerifFont of this.MainMenu.Options.Fonts;
  property;
    text "Sa&ns Serif",;
    onclick this.timeProperty,;
    checked .t.;
  custom;
    value "Sans Serif"
define menu Bold of this.MainMenu.Options;
  property;
    text "&Bold Fonts";
  custom;
    value "FontBold"
define menu BoldOff of this.MainMenu.Options.Bold;
  property;
    text "O&ff",;
    onclick this.timeProperty,;
    checked .t.;
  custom;
    value .F.
define menu BoldOn of this.MainMenu.Options.Bold;
  property;
    text "&On",;
    onclick this.timeProperty;
  custom;
    value .t.
define menu Italic of this.MainMenu.Options;
  property;
    text "&Italic Fonts";
  custom;
    value "FontItalic"
define menu ItalicOff of this.MainMenu.Options.Italic;
  property;
    text "O&ff",;
    onclick this.timeProperty,;
    checked .t.;
  custom;
    value .f.
define menu ItalicOn of this.MainMenu.Options.Italic;
  property;
    text "&On",;
    onclick this.timeProperty;
  custom;
    value .t.

endclass
function Get12Time

local time,hours

time = time()
hours = val(time)
return ltrim(str(mod(hours-1,12)+1)) + ;
      substr(time,3) + ;
      iif(hours < 12," AM"," PM")
function Get24Time

*return time()      && Time in dBASE for Windows is in 24 hour format
```

Hirdetõink:

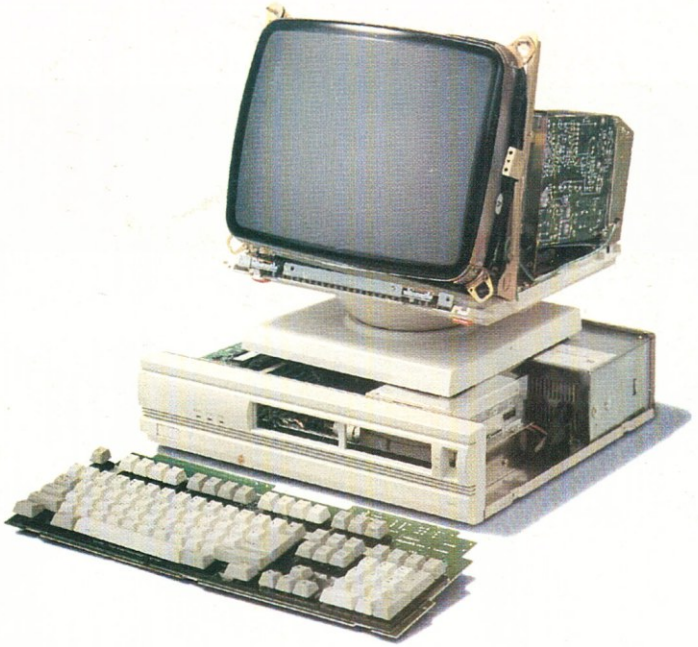
- 3M Hungaria Kft.
- Domus Komplex Vállalkozási Kft.
- StarKing Óbuda Apple Center
- LAP Stúdió
- HUMANsoft Kft.
- GAMAX Kft.
- Folio Kft.
- TEX Studio
- NASA Reklámiroda
- PC PINCE
- Partners Hungary
- DÉMA Kft.
- MTA SZTAKI
- BECO Kft.
- SYS DATA
- INTERKOMBO Kft.

Forráskód

A programozók lapja
- próbaszám

Kiadó és a kiadásért felelős:
Nagy Sándorné
Szerkesztő: Nagy Sándor
Nyomtatás: Révai Nyomda Kft.
F.v. Bánáti László
2867-94

Szerkesztőségünk címe:
1171 Bp., Szöcske u. 8.




Nord
Pol
Nördl. Polar Kreis
Aequator
Horizont