Software nr 6, lipiec 1998.
Zygmunt BOK
Zakłady Tworzyw Sztucznych 'NITRON' S.A.
ul. Zawadzkiego 1, 42-693 Krupski Młyn

OBSŁUGA BŁĘDÓW GENEROWANYCH PRZEZ NOVELL'S BTRIEVE RECORD MANAGER W APLIKACJACH BAZODANOWYCH ZORIENTOWANYCH OBIEKTOWO W ŚRODOWISKU PASCAL 7.0 TURBO VISION

Streszczenie. W artykule przedstawiono metodę, za pomocą której można zaimplementować w aplikacjach bazodanowych zorientowanych obiektowo, rozwijanych w środowisku Pascal 7.0 Turbo Vision, system obsługi błędów generowanych przez Novell's Btrieve Record Manager. System ten może być użyteczny zarówno w fazie programowania i testowania aplikacji przez programistę jak również w fazie jej eksploatacji, gdzie użytkownik końcowy jest w stanie samodzielnie reagować i skutecznie eliminować przyczyny generowania niektórych błędów.

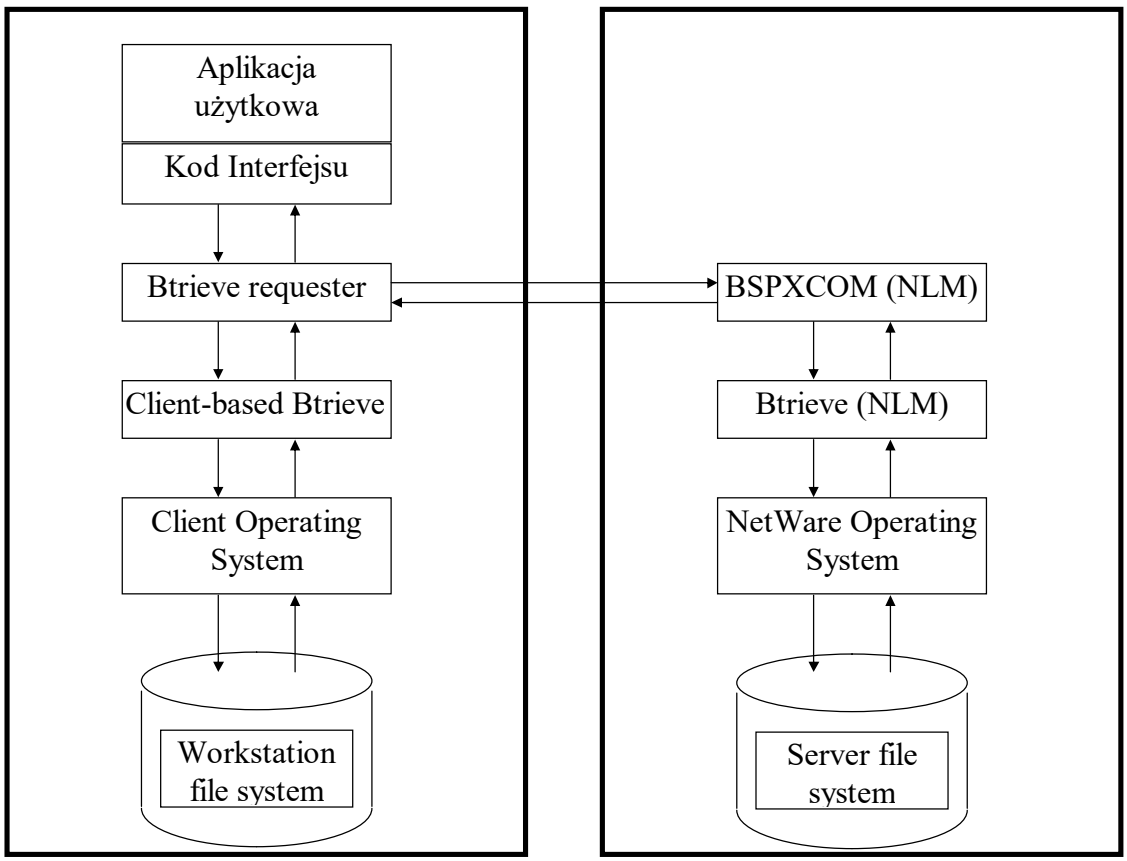
1. W P R O W A D Z E N I E

Przy tworzeniu i rozwijaniu aplikacji bazodanowych zorientowanych obiektowo typu klient-server, przy użyciu języka programowania PASCAL 7.0 TURBO VISION współpracujących z bazą danych zarządzaną przez Novell's Btrieve Record Manager, problemem który pojawia się jest kwestia wyświetlania pełnej informacji dotyczącej przyczyny generowania przez niego błędów.

Novell's Btrieve Record Manager jest systemem obsługi zbiorów dyskowych [1,2] opartych o metodę indeksowania tzw. B-drzew, z wbudowanymi mechanizmami ochrony na błędne działanie sprzętu. Zapewnia obsługę rekordów o zmiennej długości (do 64 kB) oraz rekordów o stałej długości (do 4 kB), przy użyciu max. 24 indeksów [3,5]. Umożliwia rozłożenie wolumenu na dwóch dyskach fizycznych. Może współpracować z takimi językami programowania jak: Cobol, Pascal, C, Basic.

Architektura logiczna Novell's Btrieve Record Manager'a w wersji klient-server, którą przedstawiono na rysunku 1., składa się z komponentów pozwalających aplikacji użytkowej na dostęp do danych zawartych w systemie plików zainstalowanym lokalnie na stacji roboczej lub na serwerze sieciowym. Jak widać z tego rysunku, oprócz elementów stałych istnieje również element zmienny, który jest różny dla różnych języków programowania - tzw. kodu interfejsu.

Jest to kod złączony razem z aplikacją użytkową podczas procesu kompilacji i konsolidacji wykonujący operacje I/O na plikach bazy danych, którego praktyczną implementację dla języka Turbo Pascal pokazano na wydruku 7.



Rysunek 1. Architektura logiczna systemu Novell's Btrieve Record Manager.

Aplikacje użytkowe współpracujące z bazami danych zarządzane przez Novell Btrieve Record Manager wykonują wszystkie operacje I/O na plikach tych baz danych za pomocą odpowiednich funkcji Btrieve'owskich, po wykonaniu których - w odpowiedzi - operacje te zwracają odpowiednią wartość w zmiennej statusowej typu *integer* lub innymi słowy zwracają tzw. kod statusu [4] wykonanej operacji.

Po wywołaniu określonej funkcji Btrieve'a aplikacja powinna zawsze sprawdzić wartość kodu statusu. Jeśli wartość kodu statusu równa się zero, wówczas operacja na pliku zakończyła się sukcesem, natomiast wszystkie niezerowe wartości kodu statusu wskazują na pewne błędy i aplikacja powinna sama je rozpoznać oraz odpowiednio je obsłużyć.

2. O P I S M E T O D Y

Opisywana metoda opiera się na wykorzystaniu dwóch typów obiektowych (klas obiektowych) zdefiniowanych przez f-mę Borland (znajdujących w pliku HELPFILE.PAS), tj. ***THelpViewer*** oraz ***THelpWindow***, które stanowią podstawę do

zdefiniowania dwóch nowych typów obiektowych tj., *TBtrvErrorViewer oraz TBtrvErrorWindow.* Typy te, których definicję przedstawiono na wydruku 1, mają za zadanie:

1. wydzielenie ze wstępnie przygotowanego binarnego pliku opisu błędów odpowiedniego fragmentu tekstu z opisem dotyczącym generowanego błędu, którego kontekst (numer) równy jest numerowi błędu generowanemu przez Btrieve Record Manager.
2. zaprezentowanie tego fragmentu tekstu w przewijanym oknie.

```
PBtrvErrorViewer = ^TBtrvErrorViewer;
TBtrvErrorViewer = object(THelpViewer)
   function GetPalette: PPalette; virtual;
end;

PBtrvErrorWindow = ^TBtrvErrorWindow;
TBtrvErrorWindow = object(THelpWindow)
   constructor Init(HFile: PHelpFile; Context: Word; InWhatFile: PathStr; InWhatProc: string);
   function GetPalette: PPalette; virtual;
end;
```

Wydruk 1. Definicja typów obiektowych *TBtrvErrorViewer oraz TBtrvErrorWindow,* służących do wydzielenia odpowiedniego fragmentu tekstu z opisem dotyczącym generowanego błędu oraz jego prezentacji.

Zadaniem obiektu *TBtrvErrorViewer* jest wydzielenie (ze specjalnie wcześniej przygotowanego pliku binarnego zawierającego opisy wszystkich błędów) fragmentu opisu błędu dotyczącego konkretnego numeru błędu zwracanego przez Btrieve Record Manager przy operacjach I/O na określonym pliku bazy danych i zaprezentowanie tego fragmentu w okienku zbudowanym przez obiekt *TBtrvErrorWindow.* Inicjalizacja obiektu *TBtrvErrorViewer* lub tworzenie jego instancji następuje w ciele konstruktora obiektu typu *TBtrvErrorWindow,* którego implementację przedstawiono na wydruku 2.

```
constructor TBtrvErrorWindow.Init(ErrorFile: PHelpFile; Context: Word; InWhatFile: PathStr; InWhatProc: string);
const
  TopLeftX      = 0;
  TopLeftY      = 0;
  BottomRightX = 50;
  BottomRightY = 18;
var
  R: TRect;
  HScrollBar, VScrollBar: PScrollBar;
begin
  {.......... Okno oraz  jego tytuł do prezentowania  informacji  o zaistniałym błędzie.....}
  R.Assign(TopLeftX, TopLeftY, BottomRightX, BottomRightY);
  TWindow.Init(R, 'BLAD SYSTEMU BTRIEVE', wnNoNumber);
  Options := Options or ofCentered;

  {.............. W pliku ..............................................}
  R.Assign(TopLeftX+2, TopLeftY+1, BottomRightX-2, TopLeftY+2);
  TWindow.Insert(New(PStaticText, Init(R, 'w pliku: '+InWhatFile)));

  {.............. W procedurze ..................................}
  R.Assign(TopLeftX+2, TopLeftY+2, BottomRightX-2, TopLeftY+3);
  TWindow.Insert(New(PStaticText, Init(R, 'w proc.: '+InWhatProc)));

  R.Assign(BottomRightX-3, TopLeftY+4, BottomRightX-2, BottomRightY-5);
  VScrollBar := New(PScrollBar, Init(R));
  VScrollBar^.Options := VScrollBar^.Options or ofPostProcess;
  Insert(VScrollBar);
```

```
  R.Assign(TopLeftX+2, BottomRightY-5, BottomRightX-2, BottomRightY-4);
  HScrollBar := New(PScrollBar, Init(R));
  HScrollBar^.Options := HScrollBar^.Options or ofPostProcess;
  Insert(HScrollBar);

  {..... Prezentacja treści błędu w przewijanym oknie.....}
  R.Assign(TopLeftX+2, TopLeftY+4, BottomRightX-3, BottomRightY-5);
  TWindow.Insert(New(PBtrvErrorViewer, Init(R,  HScrollBar, VScrollBar, ErrorFile, Context)));

  {..... Komentarz  .....}
  R.Assign(TopLeftX+2, BottomRightY-3, BottomRightX-2, BottomRightY-1);
  TWindow.Insert(New(PStaticText, Init(R,  ^C'   PROGRAM PRZERWANY. Odpisz numer bledu,'#13  +
                                           ^C'     oraz NACISNIJ klawisz  Esc.'#13 )));
end;
```

Wydruk 2. Implementacja konstruktora typu obiektowego *TBtrvErrorWindow.*

Tak więc opisywany typ obiektowy *TBtrvErrorWindow* ostatecznie wykorzystywany jest w procedurze ***ErrorDescription***, w ciele której następuje utworzenie jego instancji. Procedura ta wywoływana jest z poziomu poszczególnych funkcji Btrieve'owskich wykonujących operacje I/O na pewnym pliku bazy danych w sytuacji, kiedy zwracany przez te operacje kod statusu jest różny od zera. Implementację tej procedury przedstawiono na wydruku 3.

```
procedure ErrorDescription( FileErrorDescrName:  PathStr;  WhatError:  word;  InWhatFile:  PathStr;  InWhatProc: string );
var
   W           :  PWindow;
   ErrorFile   :  PHelpFile;
   ErrorStrm :  PDosStream;
begin
   ErrorStrm  := New(PDosStream, Init(FileErrorDescrName, stOpenRead));
   ErrorFile    := New(PHelpFile, Init(ErrorStrm));

   if ErrorStrm^.Status <> stOk then
        begin
           MessageBox('Nie moge otworzyc zbioru do systemu obslugi bledow.
                          Blad('+NumberToStr(ErrorStrm^.Status)+')', nil,  mfError + mfOkButton);
           Dispose(ErrorFile, Done);
        end
     else
        begin
           W := New(PBtrvErrorWindow,  Init(ErrorFile,  WhatError,  InWhatFile,  InWhatProc));
           DeskTop^.ExecView(W);
           Dispose(W, Done);
        end;
end;
```

Wydruk 3. Implementacja procedury *ErrorDescription.* Wywoływana jest z poziomu poszczególnych funkcji Btrieve'owskich wykonujących operacje I/O na pewnym pliku bazy danych.

Poniżej, na wydruku 4 przykładowo przedstawiono sposób użycia procedury *ErrorDescription* z poziomu jednej z możliwych implementacji funkcji Btrieve'a, np. *BtrvGetFirst,* służącej do zwrócenia danych z pierwszego rekordu pewnego pliku danych:

```
function BtrvGetFirst( var BtrvFile: Btrieve_File_Type;  AccesPath: integer):  boolean;
const
    Get_First = 12;
    ErrorFileName : PathStr  = 'BTRIEVE.ERR';
```

```
   InProcedure    :String    = 'BtrvGetFirst';
var
   InFile :PathStr;
begin
  BtrvGetFirst   :=  false;
  BtrvFile.KEY  := AccesPath;
  BtrvOperation(BtrvFile, Get_First);
  if Btrv_Status = 0
    then BtrvGetFirst  := true
    else begin
           InFile := BtrvFile.FileName;
           ErrorDescription(ErrorFileName, Btrv_Status, InFile, InProcedure);
         end;
end;
```

Wydruk 4. Przykład wywołania procedury *ErrorDescription* w jednej z możliwych implementacji funkcji Btrieve'a, np. *BtrvGetFirst* (pobierz pierwszy rekord z pliku bazy danych).

W tej procedurze parametr BtrvFile jest typu Btrieve_File_Type [2], o strukturze przedstawionej na wydruku 5, który przekazywany jest do funkcji *BtrvOperation* (wykonującej operacje I/O na pewnym pliku bazy danych) i służy do komunikacji pomiędzy aplikacją użytkową a plikiem bazy danych. Implementację procedury *BtrvOperation* przedstawiono na wydruku 6.

```
Btrieve_File_Type = record
                        POS          : array[1..128] of char;
                        DATA         : pointer;
                        DATALEN      : integer;
                        KBUF         : pointer;
                        KEY          : integer;
                        FileName     : PathStr;
                      end;
```

Wydruk 5. Typ rekordu służącym do komunikacji pomiędzy aplikacją użytkową a plikiem danych.

```
procedure BtrvOperation(var NameFileType:  B_File;  RodzajOperacji:  integer);
begin
  with NameFileType do
    begin
      B_Status := BTRV(RodzajOperacji,  POS,  DATA^,  DATALEN,  KBUF^, KEY);
    end;
end; {BtrvOperation}
```

Wydruk 6. Przykładowa implementacja funkcji *BtrvOperation,* wykonującej operacje I/O na pewnym pliku bazy danych.

Funkcja *BTRV(),* będąca praktyczną implementacja interfejsu firmy Novell pomiędzy Btrieve'em a aplikacją pascal'ową, wywoływana jest z poziomu procedury *BtrvOperation.* Jej zadaniem jest wykonanie właściwych operacji I/O na pewnym pliku bazy danych. Kod źródłowy tej funkcji przedstawiono na wydruku 7.

```
{*********************************************************************************************}
{  Description:        This is the Btrieve interface for Turbo Pascal (MS-DOS).                   }
{                      This routine sets up the parameter block expected by                       }
{                      Btrieve, and issues interrupt 7B.  It should be compiled                   }
{                      with the $V- switch so that runtime checks will not be                     }
{                      performed on the variable parameters.                                      }
{                                                                                                 }
{  Synopsis:           STAT := BTRV (OP, POS.START, DATA.START, DATALEN,                          }
{                                    KBUF.START, KEY);                                            }
{                      where                                                                        }
{                              OP is an integer,                                                    }
{                              POS is a 128 byte array,                                             }
{                              DATA is an untyped parameter for the data buffer,                    }
{                              DATALEN is the integer length of the data buffer,                    }
{                              KBUF is the untyped parameter for the key buffer,                    }
{                              KEY is an integer.                                                   }
{                                                                                                   }
{  Returns: Btrieve status code (see Appendix B of the Btrieve Manual).                           }
{                                                                                                   }
{  Notes:      There should NEVER be any string variables declared in the                          }
{              data or key records, because strings store an extra byte for                       }
{              the length, which affects the total size of the record.                            }
{*********************************************************************************************}
function BTRV (OP: integer;  var POS, DATA;  var DATALEN:  integer;  var KBUF;  KEY:  integer): integer;
type
   ADDR32 = record                                                {32 bit address}
            OFFSET: integer;
            SEGMENT: integer;
   end;

   BTR_PARMS = record
            USER_BUF_ADDR: ADDR32;                               {data buffer address}
            USER_BUF_LEN: integer;                               {data buffer length}
            USER_CUR_ADDR: ADDR32;                               {currency block address}
            USER_FCB_ADDR: ADDR32;                               {file control block address}
            USER_FUNCTION: integer;                              {Btrieve operation}
            USER_KEY_ADDR: ADDR32;                               {key buffer address}
            USER_KEY_LENGTH: BYTE;                               {key buffer length}
            USER_KEY_NUMBER: BYTE;                               {key number}
            USER_STAT_ADDR: ADDR32;                              {return status address}
            XFACE_ID: integer;                                   {language interface id}
   end;

var
   STAT: integer;                                          {Btrieve status code}
   XDATA: BTR_PARMS;                                       {Btrieve parameter block}
   REGS: Dos.Registers;                                    {register structure used on interrrupt call}
   DONE: boolean;

begin
   REGS.AX := $3500 + BTR_INT;
   INTR ($21, REGS);
   if (REGS.BX <> BTR_OFFSET) then          {make sure Btrieve is installed}
            STAT := 20
   else
            begin
              if (not VSet) then                    {if we haven't checked for Multi-User version}
                 begin
                        REGS.AX := $3000;
                        INTR ($21, REGS);
                        if ((REGS.AX AND $00FF) >= 3) then
                          begin
                            VSet := true;
                            REGS.AX := MULTI_FUNCTION * 256;
                            INTR (BTR2_INT, REGS);
                            MULTI := ((REGS.AX AND $00FF) = $004D);
                          end
                        else
                          MULTI := false;
                 end;

              with XDATA do                                                  {make normal btrieve call}
                 begin
                        USER_BUF_ADDR.SEGMENT := SEG (DATA);
                        USER_BUF_ADDR.OFFSET := OFS (DATA);                  {set data buffer address}
```

```
                USER_BUF_LEN := DATALEN;
                USER_FCB_ADDR.SEGMENT := SEG (POS);
                USER_FCB_ADDR.OFFSET := OFS (POS);                    {set FCB address}
                USER_CUR_ADDR.SEGMENT := USER_FCB_ADDR.SEGMENT;       {set cur seg}
                USER_CUR_ADDR.OFFSET := USER_FCB_ADDR.OFFSET+38;      {set cur ofs}
                USER_FUNCTION := OP;                                  {set Btrieve operation code}
                USER_KEY_ADDR.SEGMENT := SEG (KBUF);
                USER_KEY_ADDR.OFFSET := OFS (KBUF);                   {set key buffer address}
                USER_KEY_LENGTH := 255;                               {assume its large enough}
                USER_KEY_NUMBER := KEY;                               {set key number}
                USER_STAT_ADDR.SEGMENT := SEG (STAT);
                USER_STAT_ADDR.OFFSET := OFS (STAT);                  {set status address}
                XFACE_ID := VAR_ID;                                   {set lamguage id}
            end;

          REGS.DX := OFS (XDATA);
          REGS.DS := SEG (XDATA);

          if (NOT MULTI) then                    {MultiUser version not installed}
            INTR (BTR_INT, REGS)
          else
            begin
              DONE := FALSE;
              repeat
                REGS.BX := ProcId;
                REGS.AX := 1;
                if (REGS.BX <> 0) then
                  REGS.AX := 2;
                REGS.AX := REGS.AX + (MULTI_FUNCTION * 256);
                INTR (BTR2_INT, REGS);
                if ((REGS.AX AND $00FF) = 0) then
                  DONE := TRUE
                else begin
                  REGS.AX := $0200;
                  INTR ($7F, REGS);
                  DONE := FALSE;
                end;
              until (DONE);
              if (ProcId = 0) then
                ProcId := REGS.BX;
            end;
          DATALEN := XDATA.USER_BUF_LEN;
        end;
  BTRV := STAT;
end;
```

Wydruk 7. Praktyczna implementacja interfejsu firmy Novell pomiędzy Btrieve'em a aplikacją pascal'ową, wykonującego właściwe operacje I/O na pewnym pliku bazy danych.

Jak widać, jednym z paramertów procedury *ErrorDescription* (przedstawionego na wydruku 3) jest parametr o nazwie ***ErrorFileName*** przechowujący nazwę pliku binarnego z opisem wszystkich generowanych przez Btrieve błędów (oraz ich kontekstów), który niezbędny dla konstruktora obiektu *TBtrvErrorViewer.*
Jest to plik typu *PHelpFile* powstały w wyniku działania kompilatora TVHC.EXE (Help Compiler v. 1.0 f-my Borland) [6], którego jednym z argumentów jest plik tekstowy zawierający opisy wszystkich błędów. Przykładową strukturę tego pliku tekstowego przedstawiono na wydruku 8.

```
.topic InvalidOperation  = 1
   opis błędu ...
.topic IO_Error          = 2
  opis błędu ...
.topic NoOpen            = 3
   opis błędu ...
.topic KeyNotFound       = 4
  opis błędu ...
.topic DuplicatesError   = 5
```

```
  opis błędu ...
.
.
.
```

Wydruk 8. Przykładowa struktura pliku tekstowego zawierającego opisy wszystkich błędów zwracanych przez Btrieve Record Manager.

Każda pozycja pliku tekstowego zawierającego opisy wszystkich błędów zwracanych przez Btrieve Record Manager oznaczona jest unikalną zmienną. Liczba pozycji w tym pliku (których deklaracje zawarte w module BtrvErr przedstawiono na wydruku 9), zależy od liczby błędów zwracanych przez konkretną wersję Btrieve Record Manager'a. Wystarczy tylko wypełnić wszystkie pozycje żądaną treścią dotyczącą konkretnego błędu i poddać go kompilacji kompilatorem TVHC.EXE aby uzyskać pożądany efekt, tj. możliwość wyświetlania w przewijanym oknie fragmentu opisu błędu - co przedstawiono na rysunku 2 - którego numer równy jest niezerowemu statusowi zwracanemu przez Btrieve'a w operacjach I/O na wskazanym pliku bazy danych.

Na zakończenie, na wydruku 9 przedstawiono listing pełnego modułu (bilbioteki), który po dołączeniu do rozwijanej aplikacji bazodanowej pozwala wykorzystać wyżej opisany mechanizm wyświetlania opisów błędów w przypadku korzystania z Btrieva v.5.0.

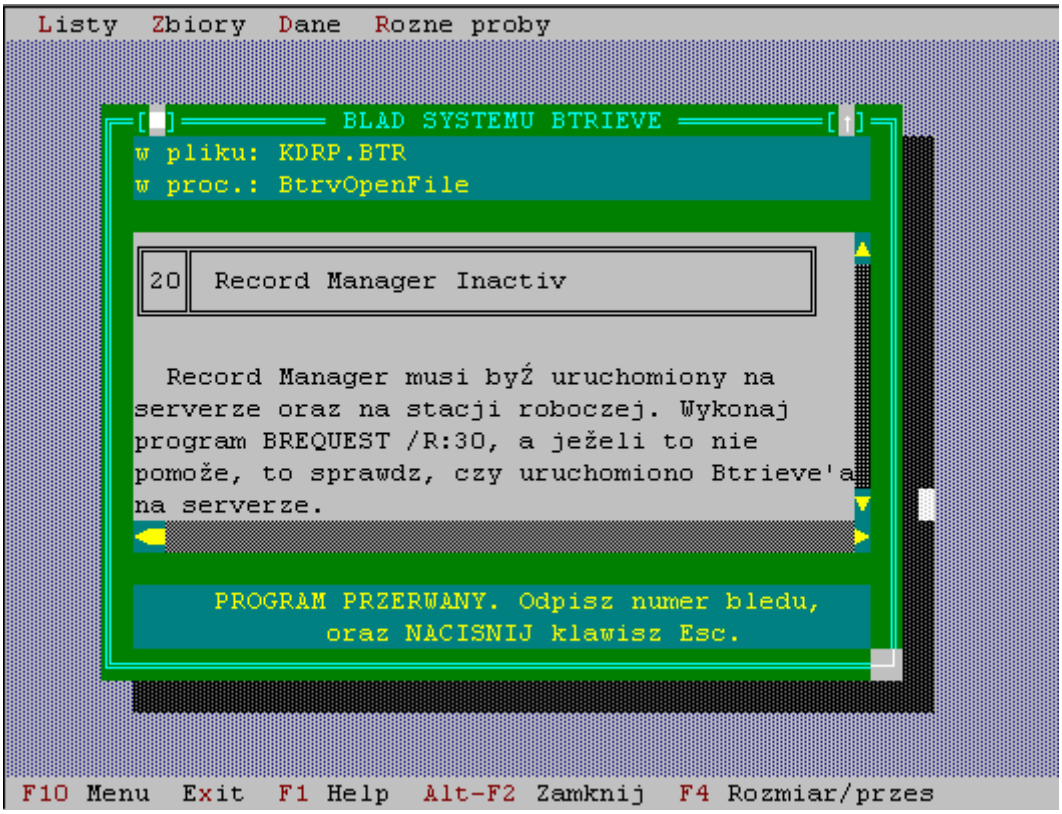
```
{************************************************************}
{                      Turbo Pascal Version  7.0             }
{                                                            }
{     Turbo Vision Unit   to link  Pascal Turbo Vision  applikations  }
{                          with Btrieve v. 5.0               }
{************************************************************}

UNIT BtrvErr;
{$F+,O+,X+,S-,D-}

INTERFACE

USES
   Dos, Objects, Drivers, Views, App, Dialogs, HelpFile, MsgBox;

CONST
 {..... Btrieve's errors .....}
 InvalidOperation                = 1;
 IO_Error                        = 2;
 NoOpen                          = 3;
 KeyNotFound                     = 4;
 DuplicatesError                 = 5;
 InvalidKeyNumber                = 6;
 DifferentKeyNumber              = 7;
 InvalidPositioning              = 8;
 EndOfFile                       = 9;
 ModifableError                  =10;

 InvalidFileName                 =11;
 FileNotFound                    =12;
 ExtensionError                  =13;
 PreOpenError                    =14;
 PreImageError                   =15;
 ExpansionError                  =16;
 CloseError                      =17;
 DiskFull                        =18;
 UnRecovarableError              =19;
 RecordManagerInactiv            =20;

 KeyBufferError                  =21;
 RecordBuffer                    =22;
```

```
PositionBlock                    =23;
PageSize                         =24;
Create_IO_Error                  =25;
NumberOfKeys                     =26;
KeyPosition                      =27;
RecordLength                     =28;
KeyLength                        =29;
BtrieveFileName                  =30;

ExtendError                      =31;
Extend_IO_Error                  =32;
ExtendName                       =34;
DirectoryError                   =35;
TransactionError                 =36;
BeginTransaction                 =37;
TransactionControlFile           =38;
End_AbortError                   =39;
TransactionMaxFiles              =40;

TransactionOperation             =41;
IncompliteAcceleratedAccess      =42;
InvalidDataRecordAddress         =43;
NullKeyPath                      =44;
InconsistentKeyFlags             =45;
AccessDeinied                    =46;
MaximumOpenFiles                 =47;
InvalidAlternateSequenceDefinition =48;
KeyTypeError                     =49;
OwnerAlreadySet                  =50;

InvalidOwner                     =51;
ErrorWritingCache                =52;
InvalidInterface                 =53;
VariablePageError                =54;
AutoIncrementError               =55;
IncompleteIndex                  =56;
Undocumented1                    =57;
CompressionBufferTooShort        =58;
FileAlreadyExist                 =59;
RejectCountReached               =60;

WorkSpaceTooSmall                =61;
IncorrectDescriptor              =62;
InvalidExtendedInsertBuffer      =63;
FilterLimitReached               =64;
IncorrectFieldOffset             =65;

AutomaticTransactionAbort        =74;
DeadLockDetected                 =78;
ProgrammingError                 =79;
Conflict                         =80;

LockError                        =81;
LostPosition                     =82;
ReadOutsideTransaction           =83;
RecordInUse                      =84;
FileInUse                        =85;
FileFull                         =86;
HandleFull                       =87;
ModeError                        =88;
DeviceFull                       =90;

ServerError                      =91;
TransactionFull                  =92;
IncopatibileLockType             =93;
PermissionError                  =94;
SessionNoLongerValid             =95;
CommunicationEnvironmentError    =96;
DataMessageTooSmall              =97;
InternalTransactionError         =98;

CBtrvErrorViewer      = #4#7#6;
CBtrvErrorWindow   = #23#42#25#46#21#21#72#71;

TYPE
```

```
PBtrvErrorViewer = ^TBtrvErrorViewer;
TBtrvErrorViewer = object(THelpViewer)
    function GetPalette: PPalette; virtual;
end;

PBtrvErrorWindow = ^TBtrvErrorWindow;
TBtrvErrorWindow = object(THelpWindow)
   constructor Init(HFile: PHelpFile; Context: Word; InWhatFile: PathStr; InWhatProc: string);
   function GetPalette: PPalette; virtual;
end;

procedure ErrorDescription(FileErrorDescrName: PathStr; WhatError: word; InWhatFile: PathStr; InWhatProc: string);

IMPLEMENTATION

{*..... TBtrvErrorWindow .....*}
constructor TBtrvErrorWindow.Init(HFile: PHelpFile; Context: Word; InWhatFile: PathStr; InWhatProc: string);
const
  TopLeftX     = 0;
  TopLeftY     = 0;
  BottomRightX = 50;
  BottomRightY = 18;
var
  R: TRect;
  HScrollBar, VScrollBar: PScrollBar;
begin
  {..... Title Error Box .....}
  R.Assign(TopLeftX, TopLeftY, BottomRightX, BottomRightY);
  TWindow.Init(R, 'BLAD SYSTEMU BTRIEVE', wnNoNumber);
  Options := Options or ofCentered;

  {..... In File .....}
  R.Assign(TopLeftX+2, TopLeftY+1, BottomRightX-2, TopLeftY+2);
  TWindow.Insert(New(PStaticText, Init(R, 'w pliku: '+InWhatFile)));

  {..... In Procedure .....}
  R.Assign(TopLeftX+2, TopLeftY+2, BottomRightX-2, TopLeftY+3);
  TWindow.Insert(New(PStaticText, Init(R, 'w proc.: '+InWhatProc)));

  {..... Error Box Interior .....}
  R.Assign(BottomRightX-3, TopLeftY+4, BottomRightX-2, BottomRightY-5);
  VScrollBar := New(PScrollBar, Init(R));
  VScrollBar^.Options := VScrollBar^.Options or ofPostProcess;
  Insert(VScrollBar);

  R.Assign(TopLeftX+2, BottomRightY-5, BottomRightX-2, BottomRightY-4);
  HScrollBar := New(PScrollBar, Init(R));
  HScrollBar^.Options := HScrollBar^.Options or ofPostProcess;
  Insert(HScrollBar);

  R.Assign(TopLeftX+2, TopLeftY+4, BottomRightX-3, BottomRightY-5);
  TWindow.Insert(New(PBtrvErrorViewer, Init(R, HScrollBar, VScrollBar, HFile, Context)));

  {..... Error Box Comments .....}
  R.Assign(TopLeftX+2, BottomRightY-3, BottomRightX-2, BottomRightY-1);
  TWindow.Insert(New(PStaticText, Init(R,
                              ^C'   PROGRAM PRZERWANY. Odpisz numer bledu,'#13 +
                              ^C'     oraz NACISNIJ klawisz Esc.'#13 )));
end;

function TBtrvErrorWindow.GetPalette: PPalette;
const
  P: String[Length(CBtrvErrorWindow)] = CBtrvErrorWindow;
begin
   GetPalette := @P;
end;


{*..... TBtrvErrorViewer .....*}
function TBtrvErrorViewer.GetPalette: PPalette;
const
   P: String[Length(CBtrvErrorViewer)] = CBtrvErrorViewer;
begin
   GetPalette := @P;
end;
```

```
procedure ErrorDescription(FileErrorDescrName: PathStr; WhatError: word; InWhatFile: PathStr; InWhatProc: string);
var
  W           : PWindow;
  ErrorFile   : PHelpFile;
  ErrorStrm:  PDosStream;
begin
  ErrorStrm  :=  New(PDosStream, Init(FileErrorDescrName, stOpenRead));
  ErrorFile   :=  New(PHelpFile, Init(ErrorStrm));
  if ErrorStrm^.Status <> stOk then
     begin
        MessageBox('Nie moge otworzyc zbioru do systemu obslugi bledow. Blad('+NumberToStr(ErrorStrm^.Status)+')',
                   nil, mfError + mfOkButton);
        Dispose(ErrorFile, Done);
     end
   else
     begin
       W := New(PBtrvErrorWindow, Init(ErrorFile, WhatError, InWhatFile, InWhatProc));
       DeskTop^.ExecView(W);
       Dispose(W, Done);
     end;
end;

end.
```

Wydruk 9. Wydruk modułu (bilbioteki), który po dołączeniu do rozwijanej aplikacji pozwala na wyświetlanie opisów generowanych błędów przez Btrieve Record Manager v.5.0.



Rysunek 2. Efekt działania obiektu ***TBtrvErrorWindow*** w aplikacjach bazodanowych przy wyświetlaniu fragmentu opisu błędu, którego numer równy jest niezerowemu statusowi zwracanemu przez Btrieve'a w operacjach I/O na wskazanym pliku bazy danych.

LITERATURA

1. RICHARD TROCINO, „Understanding Relational Theory", Novell Applications Notes, July 1993
2. KAYE PARKER, „Using NetWare SQL to Access Relational Information from Btrieve Data", Novell Applications Notes, September 1992
3. BTRIEVE PROGRAMMER'S MANUAL
4. NOVELL DEVELOPMENT PRODUCTS DIVISION, „Btrieve 386 Instalation and Operation", December 1989
5. „Btrieve Instalation and Operation" dokumentacja elektroniczna
6. BORLAND INTERNATIONAL *„Turbo Pascal 7.0"*