

```

{*****}
{                                     A R G S . P A S                                     }
{*-----*}
{ Task          : Functions for editing command line }
{               parameters                          }
{*-----*}
{ Author       : Michael Tischer / Bruno Jennrich  }
{ Developed on  : 03/20/1994                        }
{ Last update  : 4/14/1995                         }
{*****}

```

```
Unit ARGS;
```

```
Interface
```

```
Const
```

```

_char      = 0;                                { Type codes fOr GetArg }
_int       = 1;
_long      = 2;
_String    = 3;
_none      = 4;
_maxparams = 20;    { maximum number of parameters fOr NArgString }

```

```
type NArgStrings = Array[0.._maxparams-1] of string;
```

```

Function GetArg( Prefix : String;
                iType   : Integer;
                pVar    : Pointer;

```

```

        iNumEl  : Integer ) : Integer;

Function GetNArg( Prefix      : String;
                  var Strings : NArgStrings ) : Integer;

Function FindString( var Strings : NArgStrings;
                    Find      : String;
                    Number    : integer) : Integer;

Function htoi( Hstring : String;
              iDef     : Integer) : Integer;

Function strchr( SuchStr   : String;
                SearchStr : String ) : String;

Function Up( s : String ) : String;

Implementation
Uses DOS;

{ ***** }
{ Up: Convert string to uppercase letters }
{ *----- }
{ Input   : s - String to be converted }
{ Output  : The converted string }
{ ***** }
Function Up( s : String) : String;

```

```
var i : Integer;
```

```
Begin
```

```
  for i := 1 to Length( s ) do s[i] := Uppcase( s[i] );
```

```
  Up:=s;
```

```
End;
```

```
{*****}
{  GetArg : Determine command line parameters  }
{*-----}
{ Input   : Prefix - Parameter prefix (e.g.,: "-o" fOr Output) }
{           iType  - Type of parameter  (_char, _int, _long, _String }
{                   _none = tests whether parameters exist) }
{           pVar   - Address of variables which are to receive }
{                   command line parameter value. }
{           iNumEl - In case several values are separated by ',' }
{                   up to iNumElements parameters are stored in }
{                   the array specified in pVar. }
{*-----}
{ Output  : Number of determined values, or 0 if there is no }
{           command parameter available. }
{*-----}
{ Info    : To determine a string argument, the address of a string }
{           must be specified for the parameter pVar, provided }
{           iNumElements = 1. If iNumElements is greater than 1, }
{           a pointer to a variable of the NArgStrings type }
{           is expected! }
{*****}
```

```

{*****}
Function GetArg( Prefix  : String;
                 iType   : Integer;
                 pVar    : Pointer;
                 iNumEl  : Integer ):Integer;

Var i, j,
    iLen  : Integer;
    Ende  : Boolean;
    cPtr  : ^Char;
    iPtr  : ^Integer;
    Ptr   : ^Longint;
    sPtr  : ^String;
    arg   : string;
    ArgAr : ^NArgStrings;

Begin
    iLen := Length( Prefix );
    GetArg := 0;

    i := 1;
    while ( i <= ParamCount ) and
          ( Pos( Up( Prefix ), Up( ParamStr(i) ) ) <> 1 ) do
        i := i + 1;

    if i <= ParamCount then
        begin
            { Argument found }

```

```

arg := ParamStr(i);
case iType of
  _int:
    Begin
      iPtr := pVar;
      j := 0;
      Repeat
        iptr^ := 0;
        Inc( j );
        While ( arg[iLen+1] in
          ['1','2','3','4','5','6','7','8','9','0'])
          and ( iLen < length( arg ) ) do
          Begin
            Inc( iLen );
            iPtr^ := iptr^ * 10;
            iPtr^ := iPtr^ + Ord( arg[iLen]) - Ord( '0' )
          end;
        if ( ( arg[iLen+1] = ',' ) and
          ( iLen < Length( arg ) ) and
          ( j < iNumEl ) ) then
          Begin Inc(iptr); Inc(iLen); End:=FALSE; End
        else End:=TRUE;
      Until End;
      GetArg := j;
    End;

  _char:

```

```

Begin
  cPtr := pVar;
  if iLen < Length( arg ) then cPtr^ := arg[iLen+1]
  else cPtr^:=#0;
  GetArg:=1;
End;

_long:
Begin
  j := 1;
  Ptr := pVar;
  j := 0;
  Repeat
    ptr^ := 0;
    inc( j );
    while ( arg[iLen+1] in
      ['1','2','3','4','5','6','7','8','9','0'] )
      and ( iLen < Length( arg ) ) do Begin
        Inc( iLen );
        Ptr^ := Ptr^ * 10;
        Ptr^ := Ptr^ + Ord( arg[iLen] ) - Ord( '0' )
      End;
    if ( ( arg[iLen+1] = ',' ) and
      ( iLen < Length( arg ) ) and
      ( j < iNumEl ) ) then
      Begin inc( Ptr ); inc( iLen ); End:=FALSE; End
    else End:=TRUE;
  End;

```

```

    Until End;
    GetArg := j;
End;

_String:
Begin
    if iNumEl = 1 then
        begin
            sPtr := pVar;
            Repeat
                Inc( iLen );
                sPtr^ := sPtr^ + arg[iLen];
            Until iLen = Length( arg );
            GetArg:=1;
        end
    else
        begin
            ArgAr := pVar;
            j := 0;
            ArgAr^[j] := '';
            Inc( iLen );
            Repeat
                ArgAr^[j] := ArgAr^[j] + arg[iLen];
                Inc( iLen );
            until iLen < Length( arg ) then
                if arg[iLen] = ',' then
                    begin

```

```

        inc( j );
        ArgAr^[j] := '';
        inc( iLen )
    end;
    Until (iLen > Length( arg )) or (j = iNumEl);
    if(Length( ArgAr^[j] )>0) then Inc(j);
    GetArg:=j;
end;

end;

_none:
    GetArg := -1;
End;

End;

End;

{*****}
{ GetNArg : Determine all command line parameters that are not }
{   preceded by a paramater prefix character. }
{*-----}
{ Input   : Prefix   - Parameter prefix (e.g: "-o" fOr Output). All }
{           parameters that do not begin with this prefix }
{           will be retUrned. }
{           Strings  - String array in which the individual parameter }
{           strings are entered. }
{ Output  : Number of determined parameters, or 0 if no command }
{           parameters are available. }

```



```

{*****}
Function GetNArg(      Prefix    : String;
                  var Strings  : NArgStrings ) : Integer;

Var i, j, iLen : Integer;

Begin
    iLen := Length( Prefix );
    i := 1;
    j := 0;
    While ( ( i <= ParamCount ) and ( j < _maxparams ) ) do
        Begin
            if( Pos( Prefix, ParamStr(i) ) = 0 ) then
                Begin
                    Strings[j] := ParamStr(i);
                    Inc( j );
                End;
                Inc( i );
            End;
            GetNArg := j;
        End;
    End;

{*****}
{ FindString : Finds a string in a string array and returns the
  position of the found string within the array.
  *-----*
  { Input    : Strings - Array with strings to be searched
  }

```

```

{      Find      - String being searched for
      Anzahl    - Number of array entries that are filled
}
{ Output  : Position + 1 of found string in array or 0 in case
      search string is not in the array.
}
{*****}
Function FindString( var Strings : NArgStrings;
                    Find      : String;
                    Number   : integer ) : Integer;

Var i : Integer;

Begin
  FindString := 0;
  for i := 0 to Number - 1 do
    if Up( Find ) = Up( Strings[i] ) then FindString := i + 1;
  End;

{*****}
{ htoi : Converts passed Hex string to numerical value
*-----*
}
{ Input   : Hstring - String to be converted
      iDef   - Default value
}
{ Output  : Value of string or 'iDef', if the passed string doesn't
      represent a valid Hex number.
}
{*****}
Function htoi( Hstring : String; iDef : Integer) : Integer;

```

```

Var iNumChars : Integer;
    iVal,i      : Integer;

Begin
    iNumChars := 0;
    iVal := 0;
    if ( Hstring <> '' ) then
    Begin
        i := 1;
        While Hstring[i] = ' ' do Inc( i );
        While i <= Length( Hstring ) do
        Begin
            case Hstring[i] of
                '0'..'9': { Convert decimal numbers
            Begin
                iVal := iVal * $10;
                                { Move value one place to the left }
                iVal := iVal + Integer( Hstring[i] ) - Integer( '0' );
                                { Add 'Unit's' place value }
            }
                iNumChars := Succ( iNumChars );
                                { Increase number of converted characters }
            }
        End;
        'a'..'f':
        Begin
            iVal := iVal * $10;

```

```

                                { Move value one place to the left }
iVal := iVal + Integer( Hstring[i] ) - Integer( 'a' ) + 10;
                                { Add 'Unit's' place }
}

iNumChars := Succ( iNumChars );
                                { Increase number of converted characters }
End;
'A'..'F':
Begin
    iVal := iVal * $10;
                                {Move one place to the left }
    iVal := iVal + Integer( Hstring[i] ) - Integer( 'A' ) + 10;
                                { Add 'Unit's' place }
    iNumChars := Succ( iNumChars );
                                { Increase number of converted characters }
}

End;

else
Begin
    if iNumChars <> 0 then htoi := iVal else htoi:= iDef;
    Exit;
End;
End;
Inc( i );
End;
End

```

```

    else htoi := ideo;
End;

```

```

{*****}
{ strichr : Find first occurrence of a partial string without }
{ taking case into account. }
{ *-----* }
{ Input : SuchStr - string to be searched }
{ SearchStr - partial string to be searched for }
{ Output : Address of character that follows first occurrence of }
{ partial string being searched for. }
{*****}

```

```

Function strichr( SuchStr, SearchStr : String ) : String;

```

```

var iLen, i : Integer;

```

```

Begin

```

```

    strichr:='';

```

```

    if ( SuchStr <> '' ) and ( SearchStr <> '' ) then

```

```

        Begin

```

```

            iLen := Length( SearchStr );           { Length of search string }

```

```

            i := Pos( Up( SearchStr ), Up( SuchStr ) );

```

```

            if i > 0 then

```

```

                strichr := Copy( SuchStr, i+iLen, Length( SuchStr ) - i - iLen)

```

```

            End;

```

```

End;

```

```

begin

```

end.