

STRUCT.BAS 2.19 10/21/2022

D E S C R I P T I O N :

- Transforms an ASCII struct file into a list for MID\$ access.
- The result is sorted alphabetically and saved to the file "*.POS".
- A concatenation via 'CHAIN' with other programs is possible.
- This program is a component of the HISTORIC project.

R E F E R E N C E S :

9. Remember the former things of old: for I [am] God, and [there is] none else; [I am] God, and [there is] none like me,
10. Declaring the end from the beginning, and from ancient times [the things] that are not [yet] done, saying, My counsel shall stand, and I will do all my pleasure:
(Isaiah 46)
18. For verily I say unto you, Till heaven and earth pass, one jot or one tittle shall in no wise pass from the law, till all be fulfilled.
(St. Matthew 5)

The Bible

"The Bible, Authorized Version" by King James 1769, and Webster Update 1833, Oxford University Press, 1994

C O N T E N T :

- MAIN PART
- ERROR HANDLING
- SUBROUTINES VIA GOSUB
 - Recognize Structs
 - Sort Arrays
- SUBROUTINES AND FUNCTIONS
 - KeyInput\$()
 - Pause()
 - Realloc()
 - RemoveSpace\$()
 - SRealloc()

I M P O R T A N T H I N T S :

- The counting of the indices always starts by number one.
- The program expects a struct file of the following construction:
 - Struct Name: MUST BE FOUND IN THE SAME LINE BEFORE "{"
 - Struct Begin: "{"
 - Struct End: "}"
 - Array Begin: "["
 - Array End: "]"
 - Pair Attachment between "{" und "}": | : Length Name CR/LF : |
 - Maximal one pair attachment per line.
 - Comment starts and ends by line begin "-".
- Example to a struct:

comment

```

' ||      NAME { 21 }
' ||      List {
' ||          NAME   Name
' ||          8      Number
' ||          NAME   Text[ 3 ]
' ||      }
' || - Allowed are all text letters besides "{", "}", "[", "]", and "-".
' ||
' ||-----
' || H A N D L I N G :
' ||-----
' || 8/ 8/1994 - 12/19/1994      Norbert Südland, Aalen
' || 7/15/2001 -  9/11/2001      Norbert Südland, Munich
' || 8/31/2002                      Norbert Südland, Aalen
' || 4/ 5/2007 - 10/18/2007      Norbert Südland, Aalen
' || 10/13/2016 - 10/21/2022     Norbert Südland, Aalen
' ||-----

```

```

' ||-----
' || PREPARATION:
' ||-----

```

```
'OPTION EXPLICIT      'Useful with Visual Basic
```

```
CONST ConfigurationFile$ = "STRUCT.CFG"
```

```

DECLARE SUB Pause ()
DECLARE SUB Realloc (QP%, PP%(), LP%(), TermP$(), TypeP$(), Ext%)
DECLARE SUB SRealloc (QuantS%, PlaceS%(), TermS$())
DECLARE FUNCTION RemoveSpace$ (Line$, BlankPosition%, Length%)
DECLARE FUNCTION KeyInput$ ()

```

```

DIM WorkingArea$
DIM StructFile$
DIM ReturnProgram$
DIM OutputFile$
DIM Line$
DIM Name$

```

```

DIM Configuration%
DIM Struct%
DIM Comment%
DIM StructBegin%
DIM OutputFile%
DIM c%           'counter
DIM p%           'position
DIM l%           'length
DIM sQuant%
DIM Quant%
DIM Array%
DIM Current%
DIM dp%          'dummy position
DIM dl%          'dummy length
DIM Found%
DIM Limit%
DIM y%
DIM x%

```

```
DIM w%
```

```
MAIN PART:
```

```
'Preparation:
```

```
'-----'
```

```
COMMON ProgramParameter%, WorkingDirectory$, WorkingTime$, CountingMode%
ON ERROR GOTO ErrorHandler
```

```
CLS
```

```
IF WorkingDirectory$ = "" THEN
```

```
    ' As `WorkingArea$` is used the working directory, which is given by
    ' %HISTORICTEMP%, %TEMP%, or %TMP%.
    ' If (with old DOS versions) no `WorkingArea$` is given, the trial is
    ' to write onto the drive that contains the running program, too.
    ' Eventually the program stops, if the `WorkingArea$` is found to
    ' be not writable.
```

```
WorkingArea$ = ENVIRON$("HISTORICTEMP")
```

```
IF WorkingArea$ = "" THEN
```

```
    WorkingArea$ = ENVIRON$("QBASICTEMP")
```

```
    IF WorkingArea$ <> "" THEN WorkingArea$ = WorkingArea$ + "\"
```

```
10 MKDIR WorkingArea$ + "HISTORIC.TMP"
```

```
    WorkingArea$ = WorkingArea$ + "HISTORIC.TMP\"
```

```
ELSE
```

```
    IF WorkingArea$ <> "" THEN WorkingArea$ = WorkingArea$ + "\"
```

```
END IF
```

```
'Check Writability of `WorkingArea$`:
```

```
'-----'
```

```
BSAVE WorkingArea$ + "STRUKTUR.CHK", 0, 0
```

```
KILL WorkingArea$ + "STRUKTUR.CHK"
```

```
ELSE
```

```
    WorkingArea$ = WorkingDirectory$
```

```
END IF
```

```
REDIM SHARED sPos%(0)
```

```
REDIM SHARED Place%(0)
```

```
REDIM SHARED Length%(0)
```

```
REDIM SHARED sTerm$(0)
```

```
REDIM SHARED Term$(0)
```

```
REDIM SHARED Type$(0)
```

```
Configuration% = FREEFILE
```

```
1 OPEN WorkingArea$ + ConfigurationFile$ FOR INPUT AS #Configuration%
```

```
2 LINE INPUT #Configuration%, StructFile$
```

```
3 LINE INPUT #Configuration%, OutputFile$
```

```
    LINE INPUT #Configuration%, ReturnProgram$
```

```
CLOSE #Configuration%
```

```
'Grasp the Structs:
```

```
'-----'
```

```
Struct% = FREEFILE
```

```
4 OPEN StructFile$ FOR INPUT AS #Struct%
```

```
    Comment% = 0
```

```

StructBegin% = 0
sQuant% = 0
Quant% = 0
WHILE (NOT EOF(Struct%))
    LINE INPUT #Struct%, Line$
    PRINT Line$
    Line$ = RemoveSpace$(Line$, p%, 1%)
    IF LEFT$(Line$, 1) = "-" THEN
        IF Comment% = 0 THEN
            Comment% = 1
        ELSE
            Comment% = 0
        END IF
    END IF
    IF Comment% = 0 AND LEFT$(Line$, 1) <> "-" THEN
        IF LEN(Line$) > 0 THEN
            IF RIGHT$(Line$, 1) = "{" THEN
                GOSUB StructBegin
            ELSE
                IF RIGHT$(Line$, 1) = "}" THEN
                    GOSUB StructEnd
                ELSE
                    GOSUB StructContent
                END IF
            END IF
        END IF
    END IF
WEND
CLOSE #Struct%

REDIM a%(Quant%)
GOSUB Sort

PRINT "The data is written to "; OutputFile$; "."
OutputFile% = FREEFILE
OPEN OutputFile$ FOR OUTPUT AS #OutputFile%
PRINT #OutputFile%, RemoveSpace$(STR$(Quant%), p%, 1%); " data"
FOR c% = 1 TO Quant%
    PRINT #OutputFile%, RemoveSpace$(STR$(Place%(a%(c%))), p%, 1%);
    PRINT #OutputFile%, ".", ", ";
    PRINT #OutputFile%, RemoveSpace$(STR$(Length%(a%(c%))), p%, 1%), ", ";
    PRINT #OutputFile%, Term$(a%(c%)), ", "; Type$(a%(c%))
NEXT c%
CLOSE #OutputFile%

'Return to the calling program:
'-----'
IF ReturnProgram$ <> "" THEN
    REDIM Place%(0), Length%(0), Bez%(0), Typ%(0), sPos%(0), sBez%(0)
5   CHAIN ReturnProgram$
END IF

'=====
ProgramEnd:
'=====
SYSTEM
' _____ END OF THE MAIN PART _____ '

```

ERROR HANDLING:

```
'====='  
ErrorHandling:  
'====='  
  SELECT CASE ERR  
  CASE 53          'File not found  
    SELECT CASE ERL  
    CASE 1  
      PRINT "The configuration file "; ConfigurationFile$; " is missing."  
      PRINT  
      COLOR 0, 7  
      PRINT " Please start the program via HISTORIC.BAT! "  
      COLOR 7, 0  
      Pause  
      GOTO ProgramEnd  
    CASE 4  
      PRINT "The struct file "; StructFile$; " is missing."  
      PRINT  
      COLOR 0, 7  
      PRINT " Please start the program via HISTORIC.BAT! "  
      COLOR 7, 0  
      Pause  
      GOTO ProgramEnd  
    CASE 5  
      PRINT "The return program "; ReturnProgram$; " is missing."  
      PRINT  
      COLOR 0, 7  
      PRINT " Please start the program via HISTORIC.BAT! "  
      COLOR 7, 0  
      Pause  
      GOTO ProgramEnd  
    END SELECT  
  CASE 62  
    SELECT CASE ERL  
    CASE 2  
      PRINT "The configuration file "; ConfigurationFile$; " is empty."  
      PRINT  
      COLOR 0, 7  
      PRINT " Please read in the original data once again! "  
      COLOR 7, 0  
      Pause  
      GOTO ProgramEnd  
    CASE 3          'No return program is given:  
      RESUME NEXT  
    CASE ELSE  
      Pause  
    END SELECT  
  CASE 70          'Access refused  
    SELECT CASE ERL  
    CASE 10          'Writing tests onto `WorkingArea$`  
      PRINT "The working directory " + WorkingArea$  
      PRINT "is write protected. It is set at the DOS plain via " + CHR$(34);  
      PRINT "SET HISTORICTEMP=.." + CHR$(34) + "."  
      Pause
```

```

        SYSTEM
    CASE ELSE
        Pause
    END SELECT
CASE 75          'Path or File Access Error
    SELECT CASE ERL
    CASE 10      'Writing tests onto `WorkingArea$`
        RESUME NEXT
    CASE ELSE
        Pause
    END SELECT
CASE 100
    PRINT "Struct begin without name in Structur file "; StructFile$; "."
    GOTO ProgramEnd
CASE 101
    PRINT "Encapsulated struct in struct file "; StructFile$; "."
    GOTO ProgramEnd
CASE 102
    PRINT "Struct begin is missing in struct file "; StructFile$; "."
    GOTO ProgramEnd
CASE 103
    PRINT "Unknown struct name '" + Name$ + "'."
    GOTO ProgramEnd
END SELECT
ON ERROR GOTO 0
GOTO ProgramEnd
' _____ END OF THE ERROR HANDLING _____ '

'
' | SUBROUTINES VIA GOSUB: |
'
'
' | Recognize Structs: |
'
'=====
StructBegin:
'=====
    IF StructBegin% = 0 THEN
        IF 1% < 3 THEN ERROR 100
        StructBegin% = 1
        SRealloc sQuant%, sPos%(), sTerm$()
        Realloc Quant%, Place%(), Length%(), Term$(), Type$(), 1
        sTerm$(sQuant%) = LEFT$(Line$, p% - 1)
        sPos%(sQuant%) = Quant%
        Place%(Quant%) = 1
        Length%(Quant%) = 0
        Term$(Quant%) = sTerm$(sQuant%)
    ELSE
        ERROR 101
    END IF
RETURN 'StructBegin _____ '

'=====
StructEnd:
'=====

```

```

IF StructBegin% = 1 THEN
    StructBegin% = 0
    Length%(sPos%(sQuant%)) = Place%(Quant%) + Length%(Quant%) - 1
ELSE
    'Struct within one line:
    '-----'
    IF MID$(Line$, p% + 1, 1) = "{" THEN
        SRealloc sQuant%, sPos%(), sTerm$()
        Realloc Quant%, Place%(), Length%(), Term$(), Type$(), 1
        sTerm$(sQuant%) = LEFT$(Line$, p% - 1)
        sPos%(sQuant%) = Quant%
        Place%(Quant%) = 1
        Length%(Quant%) = VAL(MID$(Line$, p% + 2, 1% - p% - 2))
        Term$(Quant%) = sTerm$(sQuant%)
    ELSE
        ERROR 102
    END IF
END IF
RETURN 'StructEnd _____'

'=====
StructContent:
'=====
    Realloc Quant%, Place%(), Length%(), Term$(), Type$(), 1
    Place%(Quant%) = Place%(Quant% - 1) + Length%(Quant% - 1)

    'Array:
    '-----'
    Array% = 1
    IF RIGHT$(Line$, 1) = "]" THEN
        FOR c% = 1% - 1 TO 1 STEP -1
            IF MID$(Line$, c%, 1) = "[" THEN
                Array% = VAL(MID$(Line$, c% + 1, 1% - c%))
                1% = c% - 1
                c% = 1
            END IF
        NEXT c%
    END IF
    Term$(Quant%) = sTerm$(sQuant%) + "." + MID$(Line$, p% + 1, 1% - p%)
    Current% = Quant%
    IF Array% > 1 THEN
        Realloc Quant%, Place%(), Length%(), Term$(), Type$(), Array% - 1
        FOR c% = Array% TO 1 STEP -1
            Name$ = RemoveSpace$(STR$(c%), dp%, dl%)
            Term$(Current% + c% - 1) = Term$(Current%) + "[" + Name$ + "]"
        NEXT c%
    END IF

    'Data Length:
    '-----'
    Name$ = LEFT$(Line$, p% - 1)
    Length%(Current%) = VAL(Name$)
    IF Length%(Current%) = 0 THEN
        Found% = 0
        FOR c% = 1 TO sQuant%
            IF sTerm$(c%) = Name$ THEN
                Found% = c%
            END IF
        NEXT c%
    END IF
    'encapsulated struct

```

```

        Length%(Current%) = Length%(sPos%(Found%))
        c% = sQuant%
    END IF
NEXT c%
IF Found% = 0 THEN
    ERROR 103
ELSE
    FOR c% = 0 TO Array% - 1
        Type$(Current% + c%) = Name$
    NEXT c%
END IF
END IF
FOR c% = 1 TO Array% - 1
    Place%(Current% + c%) = Place%(Current%) + c% * Length%(Current%)
    Length%(Current% + c%) = Length%(Current%)
NEXT c%
RETURN 'StructContent _____'

```

```

' _____ '
'|                                     |'
'|                               Sort Arrays:                               |'
'|                                     |'
' _____ '

```

```

'=== '
Sort:
'=== '
    PRINT
    PRINT "Sorting..."
    FOR c% = 1 TO Quant%
        a%(c%) = c%
    NEXT c%
    FOR c% = 2 TO Quant%
        Limit% = c%
        FOR y% = 1 TO Limit% - 1
            SELECT CASE Term$(c%)
            CASE IS < Term$(a%(y%))
                FOR x% = y% TO 1 STEP -1
                    SELECT CASE Term$(c%)
                    CASE IS < Term$(a%(x%))
                        FOR w% = Limit% - 1 TO x% STEP -1
                            a%(w% + 1) = a%(w%)
                        NEXT w%
                        a%(x%) = c%
                        x% = 1
                        y% = Limit%
                    END SELECT
                NEXT x%
            END SELECT
        NEXT y%
    NEXT c%
RETURN 'Sort _____'

```

```

' _____ '
'|                                     |'
'|                               SUBROUTINES AND FUNCTIONS:                               |'
'|                                     |'
' _____ '

```

```

'===== '

```

```

FUNCTION KeyInput$
'=====
' Waits for a keyboard input and gives back the corresponding ASCII letter.
'
' Handling:
' 8/18/2001: Norbert Südland and Eckhard Walter, Adelshofen
' Check:
' 8/18/2001: Norbert Südland und Eckhard Walter, Adelshofen
'-----
DIM answer$ 'AS STRING

'Empty the keyboard buffer:
'-----
WHILE INKEY$ <> ""
WEND

'Question keyboard again, until a key has been pressed:
'-----
answer$ = ""
WHILE LEN(answer$) = 0
    answer$ = INKEY$
WEND

'Result:
'-----
KeyInput$ = answer$
END FUNCTION 'KeyInput$

'=====
SUB Pause
'=====
' Will present a statement in line 25 and wait for a pressed key.
' This pause gives the opportunity to interrupt the program and look at the
' program code.
' Under Windows XP, the command STOP not always works reliably.
'
' Handling:
' 9/ 4/2001: Norbert Südland, Munich
' 8/18/2001 Norbert Südland and Eckhard Walter, Adelshofen
' 10/10/2016: Norbert Südland, Aalen
' Check:
' 8/18/2001 Norbert Südland and Eckhard Walter, Adelshofen
' Translation:
' 2/ 5/2008: Norbert Südland, Aalen
' 10/13/2016: Norbert Südland, Aalen
'-----
DIM answer$ 'AS STRING
DIM x% 'AS INTEGER
DIM y%

'Get Current Cursor Position:
'-----
x% = CSRLIN
y% = POS(0)

'Clear Line 25:

```

```

'-----'
COLOR 7, 0
LOCATE 25, 1
PRINT SPACE$(80);

'Print Text Messages:
'-----'
LOCATE 25, 6
COLOR 0, 7
PRINT " Press any key to go on ";
COLOR 15, 0
PRINT " View Program Code: ";
COLOR 0, 7
IF ENVIRON$("COMSPEC") = "Z:\COMMAND.COM" THEN
    PRINT " [ Ctrl ] - [ Scroll ] ";
ELSE
    PRINT " [ Ctrl ] - [ Pause ] ";
END IF
COLOR 7, 0

'Wait for Pressed Key:
'-----'
answer$ = KeyInput$

'Clear Line 25:
'-----'
LOCATE 25, 1
PRINT SPACE$(80);

'Set Cursor to Old Position:
'-----'
LOCATE x%, y%

END SUB 'Pause _____'

'=====
SUB Realloc (QP%, PP%(), LP%(), TermP$(), TypeP$(), Ext%)
'=====
' Extend the memory of `PP%()`, `LP%()`, `TermP$()`, and `TypeP$()`
' by `Ext%` entries.
' The value of `QP%` is correspondingly increased.
,
' Handling:
' 9/11/2001: Norbert Südland, Munich
' 4/23/2007: Norbert Südland, Aalen
'-----

DIM c%

IF Ext% > 0 THEN

    'Save the Present Data:
    '-----'
    REDIM pl%(QP%)
    REDIM ln%(QP%)
    REDIM tm$(QP%)
    REDIM tp$(QP%)
    FOR c% = 1 TO QP%

```

```

        pl%(c%) = PP%(c%)
        ln%(c%) = LP%(c%)
        tm$(c%) = TermP$(c%)
        tp$(c%) = TypeP$(c%)
    NEXT c%

    'Extend Memory:
    '-----'
    QP% = QP% + Ext%
    REDIM PP%(QP%)
    REDIM LP%(QP%)
    REDIM TermP$(QP%)
    REDIM Type$(QP%)

    'Transmit the Present Data:
    '-----'
    FOR c% = 1 TO QP% - Ext%
        PP%(c%) = pl%(c%)
        LP%(c%) = ln%(c%)
        TermP$(c%) = tm$(c%)
        TypeP$(c%) = tp$(c%)
    NEXT c%

    'Release Memory:
    '-----'
    REDIM pl%(0)
    REDIM ln%(0)
    REDIM tm$(0)
    REDIM tp$(0)
END IF
END SUB 'Realloc _____'

'=====
FUNCTION RemoveSpace$(Line$, BlankPosition%, Length%)
'=====
' Removes all blanks except for the 1st separating blank in `Line$`.
' The position of the blank is given back in `BlankPosition%`, the
' string length in `Length%`.
'
' Handling:
' 9/11/2001: Norbert Südland, Munich
' 4/23/2007: Norbert Südland, Aalen
' 2/24/2017: Norbert Südland, Aalen
'-----
DIM counter%

'Delete Left and Right Positioned Blanks:
'-----'
Line$ = LTRIM$(RTRIM$(Line$))
Length% = LEN(Line$)

'Omit the First Blank from Deleting:
'-----'
BlankPosition% = 0
FOR counter% = 2 TO Length%
    SELECT CASE MID$(Line$, counter%, 1)
        CASE CHR$(8), CHR$(9), CHR$(255)

```

```

        MID$(Line$, counter%, 1) = SPACE$(1)
    END SELECT
    IF MID$(Line$, counter%, 1) = SPACE$(1) THEN
        IF BlankPosition% = 0 THEN
            BlankPosition% = counter%
        ELSE
            counter% = counter% - 1
            Length% = Length% - 1
            Line$ = LEFT$(Line$, counter%) + RIGHT$(Line$, Length% - counter%)
        END IF
    END IF
NEXT counter%

RemoveSpace$ = Line$
END FUNCTION 'RemoveSpace$ _____'

'=====
SUB SRealloc (QuantS%, PlaceS%(), TermS$())
'=====
' Extends the memory of `PlaceS`() and `TermS$()`, each by one entry.
' The value of `QuantS%` is correspondingly increased.
'
' Handling:
' 9/11/2001: Norbert Südland, Munich
'-----
DIM c%

'Save the Present Data:
'-----
REDIM pl%(QuantS%)
REDIM tm$(QuantS%)
FOR c% = 1 TO QuantS%
    pl%(c%) = PlaceS%(c%)
    tm$(c%) = TermS$(c%)
NEXT c%

'Extend Memory:
'-----
QuantS% = QuantS% + 1
REDIM PlaceS%(QuantS%)
REDIM TermS$(QuantS%)

'Transmit the Present Data:
'-----
FOR c% = 1 TO QuantS% - 1
    PlaceS%(c%) = pl%(c%)
    TermS$(c%) = tm$(c%)
NEXT c%

'Release Memory:
'-----
REDIM pl%(0)
REDIM tm$(0)

END SUB 'SRealloc _____'

```