```
//=======================================
// Compiler: MSL
// Project:  Zoek68KCalls
// Bestand:  Zoek68KCalls.h
// Implemen: Zoek68KCalls.cp
//=======================================

#pragma once

#include "ConsoleString.h"
#include "n1.h"
#include "z2.h"
#include "z4.h"

int main();

void Zoek68KCalls();

void Scan(rc_string pad);

void Scan(rc_string pad,
          z2 vRefNum,
          z4 dirID);

bool SelecteerBestand(pc_n1 naam);

void DoorzoekBestand(rc_string pad,
                     rc_string naam);

bool IsHoofdletter(char karakter);

bool IsIdentifierKarakter(char karakter);

void DoorzoekLijn(rc_string lijn);

void ControleerIdentifier(rc_string identifier);
```

```
//=========================================
// Compiler: MSL
// Project:  Zoek68KCalls
// Bestand:  Zoek68KCalls.cp
//=========================================

#include "Zoek68KCalls.h"

#include "Woordenboek.h"

Woordenboek woordenboek;

int main()
{
    cout << "Start van het programma.";
    cout << endl;

    cout << "Typ Command-Q om te stoppen.";
    cout << endl;

    Zoek68KCalls();

    cout << "Einde van het programma.";
    cout << endl;

    return 0;
}

void Zoek68KCalls()
{
    Scan("Project:Gewoon:Oude Unicode:");
}

void Scan(rc_string pad)
{
    cpc_char c=pad.c_str();
    Str255 p;
    CString::ConverteerCStringNaarPString(c,
                                          p.string,
                                          1,
                                          255);
    FSSpec spec;
    OSErr err=Files::OSERR_FSMakeFSSpec(0,
                                        0,
                                        p.string,
```

```
                                        spec);
    if (err!=osErr_noErr)
        {
        Exit("Pad is fout.");
        }
    z4 dirID;
    DirHulp::ZoekDirID(spec,
                       dirID);
    Scan(pad,
         spec.vRefNum,
         dirID);
}

void Scan(rc_string pad,
          c_z2 vRefNum,
          c_z4 dirID)
{
    OSErr err=osErr_noErr;
    for (z2 fDirIndex=1;;fDirIndex++)
        {
        FSSpec spec;
        spec.vRefNum=vRefNum;
        spec.parID=dirID;
        String::Init(spec.name.string);
        CInfoPBRec param;
        Blok::Zero(sizeof(CInfoPBRec),
                   &param);
        param.hFileInfo.ioNamePtr=spec.name.string;
        param.hFileInfo.ioVRefNum=vRefNum;
        param.dirInfo.ioFDirIndex=fDirIndex;
        param.dirInfo.ioDrDirID=dirID;
        err=Files::OSERR_PBGetCatInfoSync(param);
        if (err!=osErr_noErr)
            {
            if (err==osErr_fnfErr)
                {
                break;
                }
            Exit("PBGetCatInfoSync mislukt.");
            }
        char c[32];
        CString::ConverteerPStringNaarCString(spec.name.string,
                                              c);
        c_string naam=c;
        c_z1 flAttrib=param.dirInfo.ioFlAttrib;
```

```
        c_bool isMap=static_cast<bool>(flAttrib bitand flAttrib_bit4_dir);
        if (isMap)
            {
            c_string map_pad=pad+c+":";
            Scan(map_pad,
                 vRefNum,
                 param.dirInfo.ioDrDirID);
            }
        else
            {
            if (SelecteerBestand(spec.name.string))
                {
                DoorzoekBestand(pad,
                                naam);
                }
            }
        }
}

bool SelecteerBestand(cpc_n1 naam)
{
    if (String::EindigtMetCaseSensitive(naam,
                                        "\p.cp"))
        {
        return true;
        }
    if (String::EindigtMetCaseSensitive(naam,
                                        "\p.h"))
        {
        return true;
        }
    return false;
}

void DoorzoekBestand(rc_string pad,
                     rc_string naam)
{
    Invoer invoer(naam,
                  Karakter::cr,
                  pad);
    while (!invoer.Einde())
        {
        string nieuw;
        if (!invoer.LeesLijn(nieuw))
            {
```

```
            // opm: lege lijn.
            continue;
            }
        DoorzoekLijn(nieuw);
        }
}

bool IsHoofdletter(c_char karakter)
{
    if ((karakter>='A') and (karakter<='Z'))
        {
        return true;
        }
    return false;
}

bool IsIdentifierKarakter(c_char karakter)
{
    if ((karakter>='a') and (karakter<='z'))
        {
        return true;
        }
    if (IsHoofdletter(karakter))
        {
        return true;
        }
    if ((karakter>='0') and (karakter<='9'))
        {
        return true;
        }
    if (karakter=='_')
        {
        return true;
        }
    return false;
}

void DoorzoekLijn(rc_string lijn)
{
    pc_char karakter=lijn.c_str();
    bool gevonden=false;
    string identifier;
    while (*karakter)
        {
        if (!gevonden)
```

```
            {
            if (IsHoofdletter(*karakter))
                {
                gevonden=true;
                identifier=*karakter;
                }
            }
        else if (IsIdentifierKarakter(*karakter))
            {
            identifier+=*karakter;
            }
        else if (gevonden)
            {
            ControleerIdentifier(identifier);
            gevonden=false;
            }
        karakter++;
        }
    if (gevonden)
        {
        ControleerIdentifier(identifier);
        }
}

void ControleerIdentifier(rc_string identifier)
{
    if (woordenboek.Zoek(identifier))
        {
        cout << identifier;
        cout << endl;
        }
}
```

```
//=======================================
// Compiler: MSL
// Project:  Zoek68KCalls
// Bestand:  Woordenboek.h
// Implemen: Woordenboek.cp
//=======================================

#pragma once

class Woordenboek;
typedef Woordenboek* p_Woordenboek;
typedef const p_Woordenboek cp_Woordenboek;
typedef p_Woordenboek& rp_Woordenboek;
typedef const Woordenboek c_Woordenboek;
typedef c_Woordenboek* pc_Woordenboek;
typedef const pc_Woordenboek cpc_Woordenboek;
typedef Woordenboek& r_Woordenboek;
typedef c_Woordenboek& rc_Woordenboek;

class Woordenboek
{
private:
    // Globale data.
    static c_z2 aIdentifier=369;
    static cpc_char identifiers[aIdentifier];

    // Eigen types.
    struct Identifier;
    typedef Identifier* p_Identifier;
    typedef const p_Identifier cp_Identifier;
    typedef p_Identifier& rp_Identifier;
    typedef const Identifier c_Identifier;
    typedef c_Identifier* pc_Identifier;
    typedef const pc_Identifier cpc_Identifier;
    typedef Identifier& r_Identifier;
    typedef c_Identifier& rc_Identifier;

    struct Identifier
        {
        string identifier;
        p_Identifier links;
        p_Identifier rechts;
        };

    // Eigen data.
```

```
    p_Identifier m_top;

    // Constructie / Kopie / Destructie.
public:
    Woordenboek();
    ~Woordenboek();

    // Verboden.
private:
    Woordenboek(rc_Woordenboek origineel);
    void operator=(rc_Woordenboek origineel);

    // Manipulators.
    void Init();
    void VoegToe(z2 l,
                 z2 r);
    void VoegToe(pc_char identifier);
    void VoegToe(rp_Identifier top,
                 pc_char identifier);
    void MaakNieuw(rp_Identifier nieuw,
                   pc_char identifier);
public:
    bool Zoek(rc_string identifier) const;
private:
    bool Zoek(pc_Identifier top,
              rc_string identifier) const;
    void Wis(pc_Identifier top) const;
};
```

```
//=========================================
// Compiler: MSL
// Project:  Zoek68KCalls
// Bestand:  Woordenboek.cp
//=========================================

#include "Woordenboek.h"

cpc_char Woordenboek::identifiers[aIdentifier]=
    {
    "ActivatePalette",
    "AddDrive",
    "AddPt",
    "AddResource",
    "AliasDispatch",
    "Allocate",
    "AllocContig",
    "AnimateEntry",
    "AnimatePalette",
    "AttachVBL",
    "BitAnd",
    "BitClr",
    "BitMapRgn",
    "BitMapToRegion",
    "BitNot",
    "BitOr",
    "BitSet",
    "BitShift",
    "BitTst",
    "BitXOr",
    "BlockMove",
    "BlockMoveData",
    "Button",
    "CalcMenuSize",
    "Chain",
    "ChangedResource",
    "CheckItem",
    "ClearMenuBar",
    "CloseDeskAcc",
    "CloseResFile",
    "CloseWindow",
    "CommToolboxDispatch",
    "ComponentDispatch",
    "Control",
    "CopyDeepMask",
```

```
    "CopyPalette",
    "CountADBs",
    "CountMItems",
    "Create",
    "CTab2Palette",
    "Debugger",
    "DebugStr",
    "DebugUtil",
    "DECSTR68K",
    "DeferUserFn",
    "Delay",
    "Delete",
    "DeleteMCEntries",
    "DeleteMenu",
    "DelMCEntries",
    "DeltaPoint",
    "Dequeue",
    "DisplayDispatch",
    "DispMCInfo",
    "DisposCCursor",
    "DisposCIcon",
    "DisposeCCursor",
    "DisposeCIcon",
    "DisposeMCInfo",
    "DisposePalette",
    "DoVBLTask",
    "DrawMenuBar",
    "DrvrInstall",
    "DrvrRemove",
    "DTInstall",
    "Elems68K",
    "EmptyRect",
    "Enqueue",
    "EqualPt",
    "EqualRect",
    "ExitToShell",
    "FInitQueue",
    "Fix2Frac",
    "Fix2Long",
    "Fix2X",
    "FixATan2",
    "FixDiv",
    "FixMul",
    "FixRatio",
    "FixRound",
```

```
    "FlushEvents",
    "FlushFile",
    "FlushVol",
    "FP68K",
    "Frac2Fix",
    "Frac2X",
    "FracCos",
    "FracDiv",
    "FracMul",
    "FracSin",
    "FracSqrt",
    "FSDispatch",
    "Gestalt",
    "GetADBInfo",
    "GetAppParms",
    "GetCCursor",
    "GetCIcon",
    "GetCursor",
    "GetDefaultStartup",
    "GetEntryColor",
    "GetEntryUsage",
    "GetEOF",
    "GetFileInfo",
    "GetFPos",
    "GetGestaltProcPtr",
    "GetIcon",
    "GetIndADB",
    "GetItem",
    "GetItemCmd",
    "GetItmIcon",
    "GetItmMark",
    "GetItmStyle",
    "GetKeys",
    "GetMaskTable",
    "GetMCEntry",
    "GetMCInfo",
    "GetMenuBar",
    "GetMenuHandle",
    "GetMenuItemText",
    "GetMHandle",
    "GetMouse",
    "GetNewDialog",
    "GetNewMBar",
    "GetNewPalette",
    "GetOSDefault",
```

```
    "GetOSEvent",
    "GetOSTrapAddress",
    "GetPalette",
    "GetPattern",
    "GetPicture",
    "GetResInfo",
    "GetResource",
    "GetScrap",
    "GetSubTable",
    "GetToolBoxTrapAddress",
    "GetToolTrapAddress",
    "GetTrapAddress",
    "GetVideoDefault",
    "GetVol",
    "GetVolInfo",
    "GrafDevice",
    "HandAndHand",
    "HandToHand",
    "HCreate",
    "HDelete",
    "HFSDispatch",
    "HFSPinaforeDispatch",
    "HGetFileInfo",
    "HGetVInfo",
    "HGetVol",
    "HighLevelFSDispatch",
    "HiliteMenu",
    //"HiWord", // opm: is macro geworden.
    "HOpenResFile",
    "HRename",
    "HRstFLock",
    "HSetFileInfo",
    "HSetFLock",
    "HSetVol",
    "HWPriv",
    "IdleState",
    "IdleUpdate",
    "InfoScrap",
    "InitAllPacks",
    "InitEvents",
    "InitFS",
    "InitGDevice",
    "InitGraf",
    "InitMenus",
    "InitPack",
```

```
    "InitPalettes",
    "InitProcMenu",
    "InitResources",
    "InitUtil",
    "InsertMenu",
    "InSetRect",
    "InsTime",
    "InsXTime",
    "InternalWait",
    "InvalMenuBar",
    "KeyTrans",
    "KeyTranslate",
    "KillIO",
    "LoadScrap",
    "LoadSeg",
    "LodeScrap",
    "Long2Fix",
    "LongMul",
    //"LoWord", // opm: is macro geworden.
    "MakeITable",
    "MapPt",
    "MapRect",
    "MaxApplZone",
    "MemoryDispatch",
    "MemoryDispatchA0Result",
    "MenuChoice",
    "MenuSelect",
    "MethodDispatch",
    "MixedModeMagic",
    "MoveWindow",
    "NewCWindow",
    "NewGestalt",
    "NewPalette",
    "NewWindow",
    "NMInstall",
    "NMRemove",
    "NSetPalette",
    "OffSetRect",
    "OSDispatch",
    "OSEventAvail",
    "Pack1",
    "Pack10",
    "Pack11",
    "Pack12",
    "Pack13",
```

```
    "Pack14",
    "Pack15",
    "Pack2",
    "Pack3",
    "Pack4",
    "Pack5",
    "Pack7",
    "Pack8",
    "Pack9",
    "Palette2CTab",
    "PaletteDispatch",
    "PlotIcon",
    "PmBackColor",
    "PmForeColor",
    "PMgrOp",
    "PostEvent",
    "PowerDispatch",
    "PowerOff",
    "PPostEvent",
    "PrGlue",
    "PrimeTime",
    "ProtectEntry",
    "Pt2Rect",
    "PtInRect",
    "PtrAndHand",
    "PtrToHand",
    "PtrToXHand",
    "PtToAngle",
    "PutIcon",
    "Random",
    "Read",
    "ReadXPRam",
    "ReName",
    "ReplaceGestalt",
    "ReserveEntry",
    "RestoreEntries",
    "RmveResource",
    "RmvTime",
    "RstFilLock",
    "SaveEntries",
    "ScalePt",
    "ScrnBitMap",
    "SectRect",
    "SerialPower",
    "SetADBInfo",
```

```
    "SetAppBase",
    "SetApplBase",
    "SetApplLimit",
    "SetCCursor",
    "SetClientID",
    "SetDefaultStartup",
    "SetEntries",
    "SetEntryColor",
    "SetEntryUsage",
    "SetEOF",
    "SetFileInfo",
    "SetFilLock",
    "SetFilType",
    "SetFPos",
    "SetItem",
    "SetItmIcon",
    "SetItmMark",
    "SetItmStyle",
    "SetMCEntries",
    "SetMCInfo",
    "SetMenuBar",
    "SetMenuItemText",
    "SetMFlash",
    "SetOSDefault",
    "SetPalette",
    "SetPt",
    "SetRect",
    "SetResPurge",
    "SetStdCProcs",
    "SetVideoDefault",
    "SetVol",
    "SetWTitle",
    "ShieldCursor",
    "ShutDown",
    "SIntInstall",
    "SIntRemove",
    "Sleep",
    "SleepQInstall",
    "SleepQRemove",
    "SlotManager",
    "SlotVInstall",
    "SlotVRemove",
    "SlpQInstall",
    "SlpQRemove",
    "SndAddModifier",
```

```
    "SndControl",
    "SndDisposeChannel",
    "SndDoCommand",
    "SndDoImmediate",
    "SndNewChannel",
    "SndPlay",
    "SoundDispatch",
    "Status",
    "StdOpcodeProc",
    "StripAddress",
    "StuffHex",
    "SubPt",
    "SwapMMUMode",
    "SysBeep",
    "SysEnvirons",
    "SystemMenu",
    "TEActivate",
    "TEAutoView",
    "TECalText",
    "TEClick",
    "TECopy",
    "TECut",
    "TEDeactivate",
    "TEDelete",
    "TEDispatch",
    "TEDispose",
    "TEFindLine",
    "TEFindWord",
    "TEGetOffset",
    "TEGetText",
    "TEIdle",
    "TEInit",
    "TEInsert",
    "TEKey",
    "TENew",
    "TEPaste",
    "TEPinScroll",
    "TEScroll",
    "TESelView",
    "TESetAlignment",
    "TESetJust",
    "TESetSelect",
    "TESetText",
    "TEStyleNew",
    "TETextBox",
```

```
    "TEUpdate",
    "TextBox",
    "ThreadDispatch",
    "TickCount",
    "Translate24To32",
    "UnionRect",
    "UnloadScrap",
    "UnLoadSeg",
    "UnlodeScrap",
    "UpdatePixMap",
    "UpdateResFile",
    "UpdtDialog",
    "UserDelay",
    "VInstall",
    "VRemove",
    "Write",
    "WriteParam",
    "WriteResource",
    "X2Fix",
    "X2Frac",
    "ZoomWindow"
    };

Woordenboek::Woordenboek()
    : m_top(nil)
{
    Init();
}

Woordenboek::~Woordenboek()
{
    if (m_top)
        {
        Wis(m_top);
        }
}

void Woordenboek::Init()
{
    c_z2 r=aIdentifier-1;
    VoegToe(0,
            r);
}

void Woordenboek::VoegToe(c_z2 l,
```

```
                          c_z2 r)
{
    if (l>r)
        {
        return;
        }
    c_z2 m=static_cast<z2>((l+r+1)>>1);
    cpc_char identifier=identifiers[m];
    VoegToe(identifier);
    VoegToe(l,
            static_cast<z2>(m-1));
    VoegToe(static_cast<z2>(m+1),
            r);
}

void Woordenboek::VoegToe(cpc_char identifier)
{
    VoegToe(m_top,
            identifier);
}

void Woordenboek::VoegToe(rp_Identifier top,
                          cpc_char identifier)
{
    if (!top)
        {
        MaakNieuw(top,
                  identifier);
        return;
        }
    c_z4 o=top->identifier.compare(identifier);
    if (o==1)
        {
        VoegToe(top->links,
                identifier);
        return;
        }
    if (o==-1)
        {
        VoegToe(top->rechts,
                identifier);
        return;
        }
    Exit("Dubbele identifier.");
}
```

```
void Woordenboek::MaakNieuw(rp_Identifier nieuw,
                            cpc_char identifier)
{
    nieuw=new Identifier;
    nieuw->identifier=identifier;
    nieuw->links=nil;
    nieuw->rechts=nil;
}

bool Woordenboek::Zoek(rc_string identifier) const
{
    if (!Zoek(m_top,
              identifier))
        {
        return false;
        }
    return true;
}

bool Woordenboek::Zoek(cpc_Identifier top,
                       rc_string identifier) const
{
    if (!top)
        {
        return false;
        }
    c_z4 o=top->identifier.compare(identifier);
    if (o==1)
        {
        if (!Zoek(top->links,
                  identifier))
            {
            return false;
            }
        }
    else if (o==-1)
        {
        if (!Zoek(top->rechts,
                  identifier))
            {
            return false;
            }
        }
    return true;
```

```
}

void Woordenboek::Wis(cpc_Identifier top) const
{
    if (top->links)
        {
        Wis(top->links);
        }
    if (top->rechts)
        {
        Wis(top->rechts);
        }
    delete top;
}
```