```
/**
 * @file        SourceCompressor.cpp
 * @synopsis    Klassen-Definition SourceCompressor
 * @author      Jan Tammen (FH Konstanz), <jan.tammen@fh-konstanz.de>
 * @date        2005-04-14
 */

#include "SourceCompressor.h"
#include "Exception.h"

// {{{ Konstruktoren, Destruktor
// ---------------------------------------------------------------- //

/// Default-Konstruktor
SourceCompressor::SourceCompressor(void)
{
    throw FileNotReadableException("Keine Quelldatei angegeben.");
}

/// Konstruktor: Streams an Quelldatei binden
SourceCompressor::SourceCompressor (string src) :
    mSrcFilestream(src.c_str(), ios::in),
    mDestFilename(src.substr(0,src.find("."))+"_out.c"),
    mDestFilestream(mDestFilename.c_str(), ios::out),
    mOutBuffer()
{
    mSrcFilename  = src;

    if (!mSrcFilestream.good())
        throw FileNotReadableException("Kann Quelldatei nicht zum Lesen oeffnen.");
    if (!mDestFilestream.good())
        throw FileNotWriteableException("Kann Zieldatei nicht zum Schreiben oeffnen.
");

    buildControlMatrix();
}

/// Destruktor: Dateistroeme schliessen
SourceCompressor::~SourceCompressor ()
{
    mSrcFilestream.close();
    mDestFilestream.close();
}

// ---------------------------------------------------------------- //
// }}}

// {{{ Methoden
// ---------------------------------------------------------------- //

// {{{ SourceCompressor::compress ()
/// Durchfuehrung der Kompression
void SourceCompressor::compress (void)
{
    AutData currentState = STATE_BASE;
    map <char, CharGroup> CharGroupCache;

    char c;
    while (true)
    {
        this->mSrcFilestream.get(c);

        /// Zeichengruppe ermitteln, zunaechst Cache durchsuchen
        CharGroup currentCharGroup;
        map <char, CharGroup>::iterator CharGroupIt;

        if (mSrcFilestream.eof())
        {
            currentCharGroup = CGROUP_EOF;  /// EOF passt nicht in char!
```



```
        }
        else if ((CharGroupIt = CharGroupCache.find(c)) !=
                  CharGroupCache.end())
        {
            currentCharGroup = CharGroupIt->second; /// Aus Cache
        }
        else
        {
            currentCharGroup = this->findCharGroup(c);
            CharGroupCache[c] = currentCharGroup;
        }

        /// Aktion bestimmen. Falls undefiniert, Standard benutzen
        AutData currentAction =
            (controlMatrix[currentState][currentCharGroup][1] == UNDEF) ?
             controlMatrix[currentState][CGROUP_SONST][1] :
             controlMatrix[currentState][currentCharGroup][1];

        /// Folgezustand ermitteln. Falls undefiniert, Standard benutzen
        AutData nextState =
            (controlMatrix[currentState][currentCharGroup][0] == UNDEF) ?
             controlMatrix[currentState][CGROUP_SONST][0] :
             controlMatrix[currentState][currentCharGroup][0];
        nextState = (nextState == UNDEF) ? currentState : nextState;
        currentState = nextState;

        /// Aktion durchfuehren
        this->performAction(int(currentAction), c);

        /// Abbruchbedingung
        if (mSrcFilestream.eof()) break;
    }

    cerr << endl << "==> Ausgabe in Datei " << mDestFilename << " erfolgreich beende
t." << endl;
}
// }}}

// {{{ SourceCompressor::buildControlMatrix ()
/// Aufbau der Steuermatrix
void SourceCompressor::buildControlMatrix (void)
{
    /// Aufbau der Steuermatrix:
    /// controlMatrix[Zustand][Zeichen-/Zeichengruppe][0] = <Folgezustand>
    /// controlMatrix[Zustand][Zeichen-/Zeichengruppe][1] = <Aktion>
    /// Anz. Zustaende: 13, Anzahl Zeichengruppen: 12
    controlMatrix.resize(13);
    for (unsigned int state = 0; state < controlMatrix.size(); state++)
    {
        controlMatrix[state].resize(12);
        for (unsigned int chargroup = 0;
             chargroup < controlMatrix[state].size();
             chargroup++)
        {
            controlMatrix[state][chargroup].resize(2, UNDEF);
        }
    }

    // {{{ Befuellung der Steuermatrix
    // {{{ Zustand: BASE
    controlMatrix[STATE_BASE][CGROUP_BU][0]     = STATE_NAME;
    controlMatrix[STATE_BASE][CGROUP_BU][1]     = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_ZI][0]     = STATE_ZAHL;
    controlMatrix[STATE_BASE][CGROUP_ZI][1]     = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_SZ][0]     = STATE_SZF;
    controlMatrix[STATE_BASE][CGROUP_SZ][1]     = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_WZ][0]     = STATE_BASE;
    controlMatrix[STATE_BASE][CGROUP_APO][0]    = STATE_ZK;
    controlMatrix[STATE_BASE][CGROUP_APO][1]    = ACTION_STO;
```

```
    controlMatrix[STATE_BASE][CGROUP_QUOT][0]   = STATE_STRING;
    controlMatrix[STATE_BASE][CGROUP_QUOT][1]   = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_SLASH][0]  = STATE_KO__QM;
    controlMatrix[STATE_BASE][CGROUP_SLASH][1]  = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_AST][0]    = STATE_SZF;
    controlMatrix[STATE_BASE][CGROUP_AST][1]    = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_BSLASH][0] = STATE_SZF;
    controlMatrix[STATE_BASE][CGROUP_BSLASH][1] = ACTION_STO;
    controlMatrix[STATE_BASE][CGROUP_EOF][1]    = ACTION_OUT;
    // }}}

    // {{{ Zustand: ZK
    controlMatrix[STATE_ZK][CGROUP_BSLASH][0]   = STATE_ZKE;
    controlMatrix[STATE_ZK][CGROUP_SONST][0]    = STATE_ZKW;
    controlMatrix[STATE_ZK][CGROUP_SONST][1]    = ACTION_STO;
    controlMatrix[STATE_ZK][CGROUP_EOF][1]      = ACTION_OUT;
    // }}}

    // {{{ Zustand: ZKE
    controlMatrix[STATE_ZKE][CGROUP_SONST][0]   = STATE_ZKW;
    controlMatrix[STATE_ZKE][CGROUP_SONST][1]   = ACTION_STO;
    controlMatrix[STATE_ZKE][CGROUP_EOF][1]     = ACTION_OUT;
    // }}}

    // {{{ Zustand: ZKW
    controlMatrix[STATE_ZKW][CGROUP_APO][0]     = STATE_BASE;
    controlMatrix[STATE_ZKW][CGROUP_APO][1]     = ACTION_STOOUT;
    controlMatrix[STATE_ZKW][CGROUP_EOF][1]     = ACTION_OUT;
    // }}}

    // {{{ Zustand: KO?
    controlMatrix[STATE_KO__QM][CGROUP_SLASH][0]= STATE_KO;
    controlMatrix[STATE_KO__QM][CGROUP_SLASH][1]= ACTION_CLEAR;
    controlMatrix[STATE_KO__QM][CGROUP_AST][0]  = STATE_AST__KO;
    controlMatrix[STATE_KO__QM][CGROUP_AST][1]  = ACTION_CLEAR;
    controlMatrix[STATE_KO__QM][CGROUP_ZI][0]   = STATE_ZAHL;
    controlMatrix[STATE_KO__QM][CGROUP_ZI][1]   = ACTION_OUTSTO;
    controlMatrix[STATE_KO__QM][CGROUP_BU][0]   = STATE_NAME;
    controlMatrix[STATE_KO__QM][CGROUP_BU][1]   = ACTION_OUTSTO;
    controlMatrix[STATE_KO__QM][CGROUP_SONST][0]= STATE_BASE;
    controlMatrix[STATE_KO__QM][CGROUP_SONST][1]= ACTION_STOOUT;
    controlMatrix[STATE_KO__QM][CGROUP_EOF][1]  = ACTION_OUT;
    controlMatrix[STATE_KO__QM][CGROUP_SZ][0]   = STATE_SZF;
    controlMatrix[STATE_KO__QM][CGROUP_SZ][1]   = ACTION_STO;
    // }}}

    // {{{ Zustand: KO
    controlMatrix[STATE_KO][CGROUP_SONST][0]    = STATE_KO;
    controlMatrix[STATE_KO][CGROUP_EOL][0]      = STATE_BASE;
    controlMatrix[STATE_KO][CGROUP_EOF][1]      = ACTION_OUT;
    // }}}

    // {{{ Zustand: *KO
    controlMatrix[STATE_AST__KO][CGROUP_AST][0]     = STATE_KOE__QM;
    controlMatrix[STATE_AST__KO][CGROUP_SONST][0]   = STATE_AST__KO;
    controlMatrix[STATE_AST__KO][CGROUP_EOF][1]     = ACTION_OUT;
    // }}}

    // {{{ Zustand: KOE?
    controlMatrix[STATE_KOE__QM][CGROUP_SLASH][0]   = STATE_BASE;
    controlMatrix[STATE_KOE__QM][CGROUP_SONST][0]   = STATE_AST__KO;
    controlMatrix[STATE_KOE__QM][CGROUP_EOF][1]     = ACTION_OUT;
    // }}}

    // {{{ Zustand: ZAHL
    controlMatrix[STATE_ZAHL][CGROUP_ZI][0]     = STATE_ZAHL;
    controlMatrix[STATE_ZAHL][CGROUP_ZI][1]     = ACTION_STO;
    controlMatrix[STATE_ZAHL][CGROUP_SZ][0]     = STATE_SZF;
    controlMatrix[STATE_ZAHL][CGROUP_SZ][1]     = ACTION_OUTSTO;
```



```
    controlMatrix[STATE_ZAHL][CGROUP_SONST][0]  = STATE_BASE;
    controlMatrix[STATE_ZAHL][CGROUP_SONST][1]  = ACTION_OUT;
    controlMatrix[STATE_ZAHL][CGROUP_EOF][1]    = ACTION_OUT;
    // }}}

    // {{{ Zustand: SZF
    controlMatrix[STATE_SZF][CGROUP_BU][0]      = STATE_NAME;
    controlMatrix[STATE_SZF][CGROUP_BU][1]      = ACTION_OUTSTO;
    controlMatrix[STATE_SZF][CGROUP_ZI][0]      = STATE_ZAHL;
    controlMatrix[STATE_SZF][CGROUP_ZI][1]      = ACTION_OUTSTO;
    controlMatrix[STATE_SZF][CGROUP_SZ][0]      = STATE_SZF;
    controlMatrix[STATE_SZF][CGROUP_SZ][1]      = ACTION_STO;
    controlMatrix[STATE_SZF][CGROUP_WZ][0]      = STATE_SZF;
    controlMatrix[STATE_SZF][CGROUP_WZ][1]      = ACTION_NOOP;
    controlMatrix[STATE_SZF][CGROUP_APO][0]     = STATE_ZK;
    controlMatrix[STATE_SZF][CGROUP_APO][1]     = ACTION_OUTSTO;
    controlMatrix[STATE_SZF][CGROUP_QUOT][0]    = STATE_STRING;
    controlMatrix[STATE_SZF][CGROUP_QUOT][1]    = ACTION_OUTSTO;
    controlMatrix[STATE_SZF][CGROUP_SONST][0]   = STATE_BASE;
    controlMatrix[STATE_SZF][CGROUP_SONST][1]   = ACTION_OUT;
    controlMatrix[STATE_SZF][CGROUP_SLASH][0]   = STATE_KO__QM;
    controlMatrix[STATE_SZF][CGROUP_SLASH][1]   = ACTION_OUTSTO;
    controlMatrix[STATE_SZF][CGROUP_EOL][0]     = STATE_SZF;
    controlMatrix[STATE_SZF][CGROUP_EOL][1]     = ACTION_NOOP;
    controlMatrix[STATE_SZF][CGROUP_EOF][1]     = ACTION_OUT;
    // }}}

    // {{{ Zustand: NAME
    controlMatrix[STATE_NAME][CGROUP_BU][0]     = STATE_NAME;
    controlMatrix[STATE_NAME][CGROUP_BU][1]     = ACTION_STO;
    controlMatrix[STATE_NAME][CGROUP_ZI][0]     = STATE_NAME;
    controlMatrix[STATE_NAME][CGROUP_ZI][1]     = ACTION_STO;
    controlMatrix[STATE_NAME][CGROUP_SZ][0]     = STATE_SZF;
    controlMatrix[STATE_NAME][CGROUP_SZ][1]     = ACTION_OUTSTO;
    controlMatrix[STATE_NAME][CGROUP_AST][0]    = STATE_SZF;
    controlMatrix[STATE_NAME][CGROUP_AST][1]    = ACTION_OUTSTO;
    controlMatrix[STATE_NAME][CGROUP_WZ][0]     = STATE_BASE;
    controlMatrix[STATE_NAME][CGROUP_WZ][1]     = ACTION_OUT;
    controlMatrix[STATE_NAME][CGROUP_EOL][0]    = STATE_BASE;
    controlMatrix[STATE_NAME][CGROUP_EOL][1]    = ACTION_OUT;
    controlMatrix[STATE_NAME][CGROUP_EOF][1]    = ACTION_OUT;
    controlMatrix[STATE_NAME][CGROUP_SLASH][0]  = STATE_KO__QM;
    controlMatrix[STATE_NAME][CGROUP_SLASH][1]  = ACTION_OUTSTO;
    // }}}

    // {{{ Zustand: STRING
    controlMatrix[STATE_STRING][CGROUP_QUOT][0] = STATE_BASE;
    controlMatrix[STATE_STRING][CGROUP_QUOT][1] = ACTION_STOOUT;
    controlMatrix[STATE_STRING][CGROUP_SONST][0]= STATE_STRING;
    controlMatrix[STATE_STRING][CGROUP_SONST][1]= ACTION_STO;
    controlMatrix[STATE_STRING][CGROUP_EOF][1]  = ACTION_OUT;
    // }}}
    // }}}
}
// }}}

// {{{ SourceCompressor::findCharGroup ()
/// Zeichengruppe eines Eingabezeichens ermitteln
SourceCompressor::CharGroup SourceCompressor::findCharGroup (char c) const
{
    switch (c)
    {
        case '\n':
        case '\r':
            return CGROUP_EOL;
        case '\\':
            return CGROUP_BSLASH;
        case '*':
            return CGROUP_AST;
```

```
        case '/':
            return CGROUP_SLASH;
        case '\'':
            return CGROUP_APO;
        case '\"':
            return CGROUP_QUOT;
        case ' ':
        case '\t':
            return CGROUP_WZ;
        default:
            string sTemp;
            sTemp = c;

            /// Weitere Untersuchung: Buchstaben. Ziffer, Sonderzeichen
            if (isalpha(c) || c == '_')
                return CGROUP_BU;
            else if (isdigit(c))
                return CGROUP_ZI;
            else if (sTemp.find_first_of(string(":.+-=;,[](){}&<>%?~#")) != string::
npos)
                return CGROUP_SZ;

            return CGROUP_SONST;
    }

}
// }}}

// {{{ SourceCompressor::performAction ()
/// Aktion durchfuehren
void SourceCompressor::performAction (const int action, char& c)
{
    switch (action)
    {
        case ACTION_STO:        /// Zeichen in Buffer einfuegen
            mOutBuffer += c;
            break;
        case ACTION_OUT:        /// Buffer ausgeben
            mDestFilestream << mOutBuffer << endl;
            if (mShowOnScreen) cout << mOutBuffer << endl;
            mOutBuffer.clear();
            break;
        case ACTION_OUTSTO:     /// Buffer ausgeben + Zeichen speichern
            mDestFilestream << mOutBuffer << endl;
            if (mShowOnScreen) cout << mOutBuffer << endl;
            mOutBuffer.clear();
            mOutBuffer += c;
            break;
        case ACTION_STOOUT:     /// Zeichen speichern + Buffer ausgeben
            mOutBuffer += c;
            mDestFilestream << mOutBuffer << endl;
            if (mShowOnScreen) cout << mOutBuffer << endl;
            mOutBuffer.clear();
            break;
        case ACTION_CLEAR:
            mOutBuffer.clear(); /// Buffer leeren
            break;
        case ACTION_NOOP:       /// keine Aktion
        default:
            break;
    }

    return;
}
// }}}
// ---------------------------------------------------------------- //
// }}}
/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4 foldmethod=marker: */
```