Einführung in CL

Welchen besseren Weg würde es geben, sich mit CL-Programmierung vertraut zu machen, als sich eine typische CL-Prozedur anzusehen und diese in ihren Bestandteilen genauer zu betrachten?

Bestandteile einer CL-Prozedur

Unter einer Prozedur versteht man eine Sammlung von CL-Befehlen. Das Zerlegen und Analysieren der unterschiedlichen Teile eines CL-Programms eignet sich gut für die Einführung in die CL-Sprache. Die Abbildung 2.1 zeigt eine typische CL-Prozedur. Die einzelne Zeilen innerhalb einer CL-Prozedur werden auch CL-Statements oder CL-Befehle genannt. Die beiden Bezeichnungen sind synonym verwendbar.

```
PGM PARM(&greeting)
   COPYRIGHT  TEXT('(c) 2004, ZeBIS GmbH')
   DCL &greeting   *CHAR    5
   DCL &msg        *CHAR   80 VALUE(' ')
   DCL &terminal   *CHAR   10
   DCL &user       *CHAR   10
   MONMSG cpf0000
begin:
   RTVUSRPRF *CURRENT RTNUSRPRF(&user)
   RTVJOBA JOB(&terminal)
   CHGVAR &msg (&greeting                    *TCAT +
                ','                          *BCAT +
                &user                        *TCAT +
                '! Sie arbeiten an Terminal'*BCAT +
                &terminal                    *TCAT +
                '.')
   SNDPGMMSG MSG(&msg) MSGTYPE(*COMP)
ENDPGM
```

Abb. 2.1: Beispiel einer typischen CL-Prozedur

Abb. 2.2 zeigt die Struktur einer typischen CL-Prozedur. In den folgenden Abschnitten wollen wir die einzelnen Bestandteile näher betrachten und einzeln analysieren.

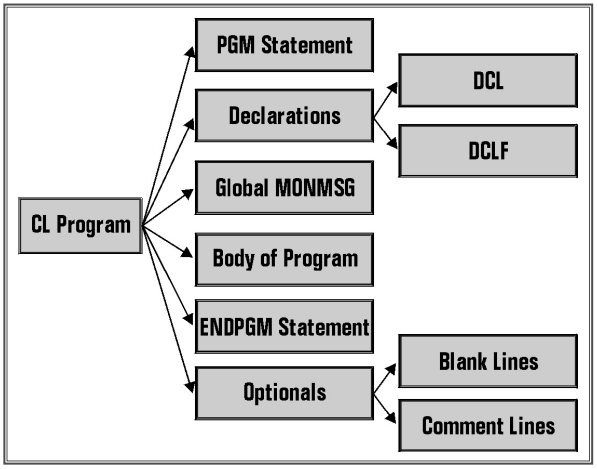

Abb. 2.2: Struktur einer typischen CL-Prozedur

Der „PGM" Befehl

Der „PGM"-Befehl (Abb. 2.3) zeigt den Beginn einer Prozedur an.

```
PGM PARM(&greeting)
```

Abb. 2.3: Ein Beispiel für den Befehl PGM

Der Parameter „PARM" enthält die Eingangs-Parameter (oder Ausgangsparameter) der Prozedur. Das Beispiel in Abb. 2.3 enthält nur einen Parameter „&GREETING". Dabei handelt es sich um eine CL-Variable.

Der „COPYRIGHT" Befehl

Der „COPYRIGHT" Befehl (Abb. 2.4) dient der Einbindung von Copyright-Angaben in einem CL-Modul. Dieser Befehl ist optional. Wenn der „COPYRIGHT" Befehl verwendet wird, muss er der Anweisung „PGM" folgen und vor MONMSG-Anweisungen und allen weiteren ausführbaren Befehlen stehen.

```
COPYRIGHT  TEXT('(c) 2004, ZeBIS')
```

Abb. 2.4: Beispiel für den COPYRIGHT-Befehl

Um die Copyright-Informationen anzuzeigen, kann das Anzeige-Modul (DSPMOD) verwendet werden.

Die Deklarationen

Abb. 2.5 zeigt ein Beispiel für die Deklaration von Programmvariablen (DCL-Befehle).

```
DCL &greeting   *CHAR    5
   DCL &msg        *CHAR   80 VALUE(' ')
   DCL &terminal   *CHAR   10
   DCL &user       *CHAR   10
```

Abb. 2.5: Beispiele für DCL-Anweisungen

Jede Deklarationsanweisung (DCL) definiert eine einzelne Variable der CL-Prozedur. Alle Variablen müssen am Anfang der Prozedur deklariert werden.

Der DCL-Befehl verfügt insgesamt über 4 Parameter, von denen die ersten beiden benötigt werden. Der erste Parameter „VAR" benennt die zu deklarierende Variable. Der zweite Parameter „TYPE" definiert, ob es sich bei der Variablen um ein Character, Decimal, Unsigned Integer, Signed Integer oder

logische Variable handelt. Der dritte Parameter „LEN“ gibt die Länge der Variablen an. Der optionale vierte Parameter „VALUE“ initialisiert die Variable – weißt dieser einen Eingangswert zu.

In Abb. 2.5 werden die Parameter „VAR“, „TYPE“ und „LEN“ so verwendet und eingegeben, dass die Prozedur übersichtlich erscheint. Sie sollten die DCLs wie folgt angeben:

```
    DCL VAR(&greeting)   TYPE(*CHAR) LEN(   5)
    DCL VAR(&msg)        TYPE(*CHAR) LEN(  80) VALUE(‘ ‘)
    DCL VAR(&terminal)   TYPE(*CHAR) LEN(  10)
    DCL VAR(&user)       TYPE(*CHAR) LEN(  10)
```

Abb. 2.6: Eine Alternative für die Codierung von DCL-Anweisungen

Beachten Sie, dass alle Variablen vom Type *CHAR sind. Neben den DCL-Anweisungen kann der Deklarations-Abschnitt der Prozedur bis zu fünf Deklarations-Dateien (DCLF) enthalten. Die Beispiele in den Abb. 2.5 und Abb. 2.6 enthalten keine Deklarations-Datei-Anweisungen (DCLF).

MONMSG

Obwohl der Befehl „Nachrichten-Überwachung“ (MONMSG) ausführlich im Kapitel 6 beschrieben wird, wollen wir an dieser Stelle kurz auf diesen eingehen. Ein Beispiel in Abb. 2.7 zeigt die Verwendung von „MONMSG“.

```
MONMSG cpf0000
```

Abb. 2.7: Beispiel eines MONMSG-Befehls

Die MONMSG-Anweisung wird dazu verwendet, um im Falle einer fehlerhaft ausgeführten CL-Prozedur oder eines fehlerhaft ausgeführten CL-Befehls, korrigierend einzugreifen.

Wenn MONMSG zwischen den Deklarationsanweisungen und der eigentlichen Prozedur angegeben wird, wirkt sie als allgemeine Fehlerbehandlung für die gesamte Prozedur. Wird die Anweisung MONMSG woanders in dem CL-Programm angegeben, so bezieht sie sich lediglich auf den direkt vorangegangenen Befehl.

Der Prozedur-Hauptteil

Abb. 2.8 zeigt ein Beispiel einer typischen CL-Prozedur

```
Beginn:
RTVUSRPRF *CURRENT RTNUSRPRF(&user)
   RTVJOBA JOB(&terminal)
   CHGVAR &msg (&greeting                    *TCAT +
                ','                          *BCAT +
                &user                        *TCAT +
                '!Sie arbeiten an Terminal'  *BCAT +
                &terminal                    *TCAT +
                '.')
   SNDPGMMSG MSG(&msg) MSGTYPE(*COMP)
```

Abb. 2.8: Beispiel eines Prozedur-Hauptteils

Der Prozedur-Hauptteil ist der Bereich, in dem die Aktionen ausgeführt werden. Es ist das A und O der CL-Prozedur. Obwohl die Befehle „PGM“ und „DCL“ in den meisten CL-Prozeduren benötigt werden, führen sie keine Aktionen aus. Die Befehle innerhalb einer Prozedur unterscheiden sich grundsätzlich davon.

In dem vorhergehenden Beispiel (Abb. 2.8) ist der erste Befehl in der Hauptprozedur der Befehl “Benutzerprofil auffinden“ (RTVUSRPRF), der der Markierung „begin:“ vorausgeht.

Markierungen sind nur dann erforderlich, wenn sie zu einem Ziel einer „GOTO“, „LEAVE“ oder „ITERATE“ Anweisung gehören. Die Verwendung solcher Markierungen ist dem Entwickler selbst überlassen. In diesem Beispiel verweist die Markierung „begin:“ auf den ersten ausführbaren Befehl des Hauptteils der Prozedur.

Die Anweisung „ENDPGM“

Die Anweisung „ENDPGM“ zeigt das Ende der Prozedur an. Der CL-Compiler ignoriert den Quellcode, der sich hinter der ENDPGM-Anweisung befindet. Wenn Sie sich dazu entscheiden, keine ENDPGM-Anweisung zu verwenden, wird Ihre Prozedur beim Erreichen des Endes der Befehlsanweisungen des Prozedur-Haupteiles beendet. Die Verwendung von ENDPGM macht den Quellcode übersichtlicher und vermeidet Unklarheiten darüber, wo die Prozedur endet. Ein Beispiel dazu finden Sie in Abb. 2.9.

```
ENDPGM
```

Abb. 2.9: Beispiel eines ENDPGM-Befehls

Source-Code Eingabe mit SEU

Bevor Sie SEU zur Eingabe eines CL-Source-Codes starten, wählen Sie eine Bibliothek aus, in der der Code gespeichert werden soll. Stellen Sie sicher, dass sich in dieser Bibliothek eine physische Quelldatei für das Speichern des CL-Source-Codes befindet.

Die physische Quelldatei

Im IBM Standard wird jeder CL-Souce-Code in einer physischen Quelldatei „QCLSRC“ gespeichert. Dieser Name ist nicht zwingend vorgeschrieben. Sie können Ihre physische Quelldatei auch zum Beispiel „CLSOURCE“, „SOURCE_CL“ oder „FRED“ nennen. Es macht wirklich keinen Unterschied: In jedem Fall haben die Umwandlungsbefehle für CL-Source-Teildateien als Standardwert QCLSRC in dem Parameter für die physische Quelldatei. Deshalb macht die Verwendung von QCLSRC durchaus Sinn und spart Zeit.

Wenn Sie keine QCLSRC-Datei haben, so können Sie diese wie in Abb. 2.10 gezeigt, erstellen. „MYLIB“ steht für den Namen der Bibliothek, in die Sie die physische Quelldatei stellen.

```
CRTSRCPF FILE(MYLIB/QCLSRC) TEXT('CL source')
```

Abb. 2.10: Erstellen einer Quell-Teildatei QCLSRC

Starten von SEU

Sie müssen den Quellcode für die Prozedur mit SEU erfassen. Entweder kann SEU manuell oder aus dem Program Development Manager (PDM) gestartet werden. Um SEU manuell zu starten, geben Sie „STRSEU“ in der Befehlszeile ein und drücken Sie die Taste „F4“. Dann erscheint eine Anzeige wie in Abb. 2.11 gezeigt.

Geben Sie die Parameter-Werte wie in der Abbildung dargestellt ein und drücken Sie die „Enter“-Taste.

Um SEU aus dem PDM heraus zu starten, geben Sie den Befehl „Mit Teildateien in PDM arbeiten“ (WRKMBRPDM) ein und drücken Sie die „F6“-Taste um eine neue Teildatei zu erstellen.

```
                    Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file  . . . . . .   qclsrc____    Name, *PRV

Library . . . . . .        mylib_____  Name, *LIBL, *CURLIB, *PRV

Source member  . . . . .    hello_____    Name, *PRV, *SELECT

Source type  . . . . . .   clle______    Name, *SAME, BAS, BASP, C...

Option . . . . . .      *BLANK____    *BLANK, ' ', 2, 5, 6

Text 'description' . . . . . .  Example_from_Chapter_Two_____________________

                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

Abb. 2.11: Die STRSEU-Anzeige

Anpassen einer Anweisung mit „F4“

CL-Prozeduren bestehen aus „Freiformat“- (Free-Format) Anweisungen. Anders als bei der Verwendung von DDS oder RPG ist der Code nicht an bestimmte Stellen, Spalten oder Zeilen innerhalb der Quell-Teildatei gebunden. Dies gibt eine gewisse Freiheit, den Code in einem individuellen Stil zu schreiben. In CL hindert Sie nichts daran, den Source-Code einer „HELLO“ Prozedur wie in Abb. 2.12 gezeigt einzugeben.

```
PGM &GREETING
DCL &GREETING *CHAR 5
DCL &USER *CHAR 10
DCL &TERMINAL *CHAR 10
DCL &MSG *CHAR 80 ' '
MONMSG CPF0000
BEGIN: RTVUSRPRF *CURRENT &USER
RTVJOBA &TERMINAL
CHGVAR &MSG (&GREETING *TCAT ',' *BCAT &USER *TCAT '!  Sie arbeiten an +
Terminal' *BCAT &TERMINAL *TCAT '.')
SNDPGMMSG MSG(&MSG) MSGTYPE(*COMP)
ENDPGM
```

Abb. 2.12: Beispiel eines CL-Codes ohne Formatierung

Sie können den CL-Code auf diese Weise eingeben, wenn Ihnen das so gefällt. Der Compiler unterscheidet nicht zwischen einer „einfachen" und „schwer zu lesenden" Eingabe. Der Compiler wird den Code in jedem Fall gleich umsetzen und die Prozedur wird unabhängig von der Art der Kodierweise ausgeführt werden.

Da der Mensch einen Hang zu Ästhetik hat, neigt er auch dazu, einen Source-Code zu bevorzugen, dessen Erscheinungsbild ansprechend ist und der leicht zu lesen ist. Ein schwer zu lesender Source-Code wird hingegen weniger gerne verwendet. Andererseits können Sie auch die „F4"-Taste und die „Enter"-Taste in jeder Zeile verwenden.

Sehen Sie sich dazu die Abb. 2.13 an:

```
PGM        PARM(&GREETING)
             DCL        VAR(&GREETING) TYPE(*CHAR) LEN(5)
             DCL        VAR(&USER) TYPE(*CHAR) LEN(10)
             DCL        VAR(&TERMINAL) TYPE(*CHAR) LEN(10)
             DCL        VAR(&MSG) TYPE(*CHAR) LEN(80)
             MONMSG     MSGID(CPF0000)
 BEGIN:      RTVUSRPRF  USRPRF(*CURRENT) RTNUSRPRF(&USER)
             RTVJOBA    JOB(&TERMINAL)
             CHGVAR     VAR(&MSG) VALUE(&GREETING *TCAT ',' *BCAT +
                          &USER *TCAT '!  Sie arbeiten an Terminal' +
                          *BCAT &TERMINAL *TCAT '.')
             SNDPGMMSG  MSG(&MSG) MSGTYPE(*COMP)
             ENDPGM
```

Abb. 2.13: CL-Code mit SEU Bedienerführung

Die meisten Entwickler bevorzugen das automatische Formatieren durch die Befehlsunterstützung (diese wird durch das Drücken der „F4"-Taste beim Öffnen des Quellcodes aktiviert). Der Quellcode in Abb. 2.13 wurde durch die Befehlsunterstützung formatiert.

Alles was Sie tun müssen, um die Bedienerführung verwenden zu können, ist die Eingabe eines Befehls irgendwo in der Quelldatei und anschließend das Betätigen der „F4"-Taste. Nun können Sie die erforderlichen Eingaben vornehmen. Durch die Angabe der Quelldatei-Art „CLLE" wird der Inhalt der Bedienerführung (F4) gesteuert. Durch die Angabe „CLLE" wird die Syntax-Prüfung und die Bedienerführung für die CL-Programmierung aktiviert. Sie können den Befehl auch im Free-Format eingeben (inklusive der Parameter mit/ohne der dazugehörigen Schlüsselworte) und anschließend die „F4"-Taste drücken.

Die Bedienerführung wandelt zum Beispiel eine Eingabe des Befehls „CHGVAR“, der wie in Abb. 2.14 eingegeben wird, in das Format in Abb. 2.15 um.

```
CHGVAR &MSG (&GREETING *TCAT ‘,’ *BCAT &USER *TCAT ‘!  +
Sie arbeiten +

An Terminal’ *BCAT &TERMINAL *TCAT ‘.’)
```

Abb. 2.14: Code Beispiel Formatierung ohne Bedienerhilfe

```
CHGVAR    VAR(&MSG) VALUE(&GREETING *TCAT ‘,’ *BCAT +

              &USER *TCAT ‘!  Sie arbeiten an Terminal’ +

              *BCAT &TERMINAL *TCAT ‘.’)
```

Abb. 2.15: Formatierter Code mit Verwendung Bedienerführung

Die Bedienerführung übernimmt folgende Aufgaben:

- Beginnt die Eingabe in der Spalte 2.
- Setzt den Beginn des Befehlsnamens in Spalte 14.
- Definiert den Beginn der Parameter in Spalte 25.
- Fügt allen Parametern die entsprechenden Schlüsselworte zu.
- Rückt alle Fortsetzungszeilen um 2 Stellen nach rechts ein und beginnt diese in Spalte 27.
- Fügt Zeilen hinzu bzw. entfernt Zeilen, um die Darstellung des Befehls zu optimieren – dabei gilt als äußerste Begrenzung die Spalte 71.
- Setzt alle Bezeichnungen, Befehlsnamen und Schlüsselworte in Großbuchstaben um, wenn diese in Kleinbuchstaben erfasst werden.
- Setzt in Zeichenfeldern alle Einzelwörter in Großbuchstaben um, wenn diese nicht in Hochkommata eingebettet sind.

- Setzt automatisch Hochkommata bei Verwendung von Multi-Wort-Zeichenfeldern.
- Verdoppelt alle eingeschlossenen einzelnen Hochkommata.

Fortgeschrittene CL-Programmierer verwenden in der Regel keine Bedienerführung (Sie in Zukunft vielleicht auch nicht), denn: Wenn Sie auf die Verwendung der Bedienerführung verzichten, können Sie selbst über den Aufbau und die Struktur Ihrer Programme entscheiden.

Groß- oder Kleinschreibung?

Der CL-Compiler unterscheidet nicht zwischen der Groß- oder Kleinschreibung eingegebener Befehle, Schlüsselworte oder Parameter. Dasselbe gilt auch für den Quellcode-Compiler. Es bleibt also Ihnen selbst überlassen, wie Sie die Eingabe vornehmen – in Groß- oder in Kleinbuchstaben.

Da die Bedienerführung die in Kleinbuchstaben eingegebenen Befehle in Großbuchstaben umsetzt, empfiehlt sich auch die Direkteingabe in Großbuchstaben. Wenn Sie die Bedienerführung nicht verwenden, dann ist es Ihnen überlassen. Sie können den Befehl „SET CAPS OFF“ auf der SEU-Befehlszeile ausführen, um die Eingabe in Kleinbuchstaben zu belassen oder durch den Befehl „SET CAPS ON“ alle Eingaben in Großbuchstaben umzusetzen. Der „HELLO“ Quellcode erscheint ungewohnt, wenn die Eingaben in Kleinbuchstaben erfolgt sind – siehe Abb. 2.16.

```
pgm        parm(&greeting)
             dcl        var(&greeting) type(*char) len(5)
             dcl        var(&user) type(*char) len(10)
             dcl        var(&terminal) type(*char) len(10)
             dcl        var(&msg) type(*char) len(80)
             monmsg     msgid(cpf0000)
 begin:      rtvusrprf  usrprf(*current) rtnusrprf(&user)
             rtvjoba    job(&terminal)
             chgvar     var(&msg) value(&greeting *tcat ',' *bcat +
                          &user *tcat '!  Sie arbeiten an Terminal' +
                          *bcat &terminal *tcat '.')
             sndpgmmsg  msg(&msg) msgtype(*comp)
             endpgm
```

Abb. 2.16: Beispiel eines Source-Codes in Kleinbuchstaben

Positionsgebundene Parameter oder Schlüsselworte?

Befehlsparameter können auf zwei unterschiedliche Art und Weisen eingegeben werden: Positionsgebunden oder mit Schlüsselworten.

Bei der manuellen Eingabe wird häufig die positionsgebundene Eingabe von Parametern bevorzugt, da diese schneller einzugeben sind. Wie auch immer die Eingabe erfolgt – niemand muss einen eingegebenen Befehl lesen können als der Interpreter!

Wenn Sie Befehle in einer CL-Quelldatei eingeben, wird dieser gleich ausgeführt – unabhängig ob Sie Schlüsselworte angegeben haben oder nicht. Wie auch immer – werden Schlüsselworte angegeben, so erleichtert dies die Lesbarkeit der Statements des Quellcodes. Die Entscheidung über die Verwendung der Schlüsselworte liegt also bei Ihnen.

Um Ihnen einen Vergleich zu ermöglichen, schauen Sie sich die Abb. 2.17 an: In der ersten Version wurde der Befehl „Doppeltes Objekt erstellen“ (CRTDUPOBJ) ohne Schlüsselworte angegeben. In der zweiten Ausführung sind Schlüsselworte enthalten. Die zweite Version ist dadurch leichter lesbar.

```
CRTDUPOBJ PGM1 MYLIB *PGM YOURLIB PGM1A
CRTDUPOBJ OBJ(PGM1) FROMLIB(MYLIB) OBJTYPE(*PGM) +
          TOLIB(YOURLIB) NEWOBJ(PGM1A)
```

Abb. 2.17: Beispiel des Befehls CRTDUPOBJ mit Schlüsselworten

Schaut man sich einige der häufig verwendeten Befehle an – zum Beispiel „Variable ändern“ (CHGVAR), die nur zwei Parameter enthalten, liegt es wieder an dem Programmierer, welche Version der Eingabe er bevorzugt. Abb. 2.18 zeigt zwei Versionen (mit und ohne Schlüsselworte).

```
CHGVAR &USER 'QSYSOPR'
CHGVAR VAR(&USER) VALUE('QSYSOPR')
```

Abb. 2.18: Beispiel für CHGVAR mit Schlüsselworten

In diesem Beispiel sind beide Versionen leicht zu interpretieren und zu lesen. Ein anderes Beispiel ist der Befehl „GOTO“, der in der Abb. 2.19 enthalten ist.

```
GOTO CONTINUE
GOTO CMDLBL(CONTINUE)
```

Abb. 2.19: Beispiel für GOTO mit Schlüsselworten

Das Schlüsselwort „CMDLBL“ dient in diesem Fall nicht wirklich der Erleichterung der Lesbarkeit des Codes.

Der Art und Weise wie der CL-Source-Code eingegeben wird, liegt also im Ermessen des Entwicklers oder den Programmiervorgaben des Unternehmens. Im Anhang „B“ werden einige weitere Beispiele im Detail behandelt und beschrieben.

In neuer Zeile fortsetzen

Wenn ein CL-Befehl mehr Parameter enthält, als in eine Code-Zeile passen, dann muss die Eingabe in einer Folgezeile (oder mehrerer Folgezeilen) fortgesetzt werden. Dies wird in CL durch das Einfügen der Zeichen „+“ oder „-“ ermöglicht.

Wenn Sie die Bedienerführung („F4“-Taste) verwenden, werden Zeilenumbrüche automatisch durchgeführt. Ebenfalls werden die Trennzeichen bei Bedarf eingefügt. Wenn Sie keine Bedienerführung verwenden, müssen Sie selbst die Zeilenfortsetzungszeichen einfügen.

Der Unterschied in der Verwendung der Zeilenfortsetzungszeichen „+“ und „-“ besteht in der Interpretation der Fortsetzungszeile durch den Compiler. Bei der Verwendung von „+“ ignoriert der Compiler alle führenden Leerstellen in der nächsten Zeile. Die Fortsetzungszeile beginnt somit erst mit dem ersten Zeichen. Verwenden Sie das Zeilenfortsetzungszeichen „-“ so werden auch die in der Fortsetzungszeile enthaltenen führenden Leerstellen mit berücksichtigt.

Diese Differenzierung ist nur dann von Bedeutung, wenn Sie eine Anweisung irgendwo in einer Zeichenkostanten trennen. Das Beispiel in Abb. 2.20 soll dies verdeutlichen:

```
CHGVAR VAR(&X) VALUE('AB +
          CD')
```

Abb. 2.20: Beispiel einer Zeilentrennung mit „+“

Die Variable „&X“ enthält den Wert „AB CD“ („AB“ ist durch eine Leerstelle von „CD“ getrennt), da in der Quellangabe eine Leerstelle zwischen „B“ und dem „+“ Zeichen eingegeben wurde. Alle führenden Leerstellen werden in der nächsten Zeile ignoriert. Abb. 2.21 zeigt denselben Befehl, allerdings wird hier anstelle des „+“ das „-“ Zeichen verwendet.

```
CHGVAR VAR(&X) VALUE('AB -
          CD')
```

Abb. 2.21 Beispiel einer Zeilentrennung mit „-”

In diesem Fall wird der Inhalt der Variablen „&X“ so aussehen: „AB CD“ (zehn Leerstellen zwischen „AB“ und „CD“). Zwischen „B“ und dem „-“ Zeichen ist lediglich eine Leerstelle angegeben worden. In der Folgezeile sind 9 führende Leerstellen enthalten.

Einrücken oder nicht einrücken?

Die Vorteile der Free-Format-Programmierung in CL können durch Einrücken des Codes erweitert werden. Wenn Sie zum Beispiel DO/ENDDO-Schleifen einrücken, so wird der Code übersichtlicher und auch die einzelnen Quellcode-Anweisungen lassen sich besser interpretieren. Je komplexer Ihre Anweisungen werden, umso nützlicher ist die Funktion der Quellcode-Einrückung.

Nebenbei bemerkt: Bei der Verwendung der Bedienerführung (F4) zur Formatierung des Codes werden alle Einrückungen ignoriert und überschrieben. Wie bereits zuvor erwähnt, wird durch die Bedienerführung der Beginn der Anweisungen im Quellcode auf die Position 14 gesetzt. Parameter beginnen grundsätzlich auf Position 25.