2 Kapitel

Einführung in Free-Format RPG IV

Einführung in Free-Format RPG IV

Einführung in Free-Format RPG IV

Mit der Verfügbarkeit der Version V5R1 des WebSphere Development Studio – der Tool-Suite von IBM für die Anwendungsentwicklung der iSeries – wurde den RPG-Anwendungsentwicklern erstmals die Möglichkeit geboten, RPG Programme im Free-Format zu erstellen. Dieser Ankündigung über die Verfügbarkeit des Free-Formats wurde von Seiten der IBM Kunden allerdings nur wenig Aufmerksamkeit geschenkt. Viele Programmierer, mit denen ich zu dieser Zeit gesprochen habe, zeigten nur ein geringes Interesse an dem neuen Konzept und der Idee des Free-Format RPG IV.

Es waren nur wenige RPG IV Programmierer, die mich um Hilfe baten – das Interesse an dieser neuen und modernen Möglichkeit der Anwendungsentwicklung mit RPG war verschwindend gering. Einzelne ambitionierte Programmierer, die sich selbst und der neuen Technik eine Chance gegeben haben, fanden dagegen sehr schnell heraus, dass die rasch erzielten Ergebnisse eine Bestätigung des eigenen Engagements und der Vision der IBM waren. Allerdings gab es in der ersten Version des Free-Format RPG noch Einschränkungen in der Funktionsfähigkeit. In vielen Bereichen konnte nicht der gesamte Code in Free-Format geschrieben werden, sondern musste vielmehr durch Fix-Format-Code ergänzt werden (Beispiele dafür sind KLIST-Anweisungen oder auch KFLD-Definitionen). Durch diesen Mix der Free-Format Anweisungen und der Fix-Format-Codierung war das gesamte Erscheinungsbild der Free-Format-Programme klobig und alles andere als leicht lesbar. Sicher mit einer der Gründe für die anfangs stark ablehnende Haltung vieler Anwendungsentwickler gegenüber dieser ersten Version.

Mit dem Erscheinen der Version V5R2 hat IBM viel in die Verbesserung und die Weiterentwicklung des Free-Format- RPG

IV investiert. So lassen sich seit dieser Version zum Beispiel auch Schlüssellisten oder Schlüsselfelder im Free-Format codieren. Auch die Verwendung von Schlüsseln in CHAIN oder SETxx-Anweisungen ist in dieser Version endlich realisiert.

Der Einsatz der Built-In Function %Kds trug ebenfalls stark zu der wesentlich verbesserten Akzeptanz bei. Mit diesen Neuerungen hat IBM einen entscheidenden und auch notwendigen Schritt in Richtung Einsatzfähigkeit und Akzeptant des Free-Format RPG unternommen.

Andere Neuerungen, die mit V5R2 eingeführt wurden – hier sei als Beispiel die Verwendung des „+“ Operanten zu nennen, mit dem jetzt auch Verkettung bzw. Erweiterungen von bestimmten Operanten möglich sind – lassen einen direkten Vergleich der RPG-Free-Format-Prozeduren zu C- und Java Prozeduren zu. Weitere neue Built-In Functions erweitern zudem die Möglichkeit der Modernisierung der RPG IV Prozeduren und stellen einen Meilenstein in Richtung der Modernisierung dieser Programmiersprache dar.

Heute bietet RPG IV also durchaus dieselben Methoden und Techniken der Anwendungsentwicklung, wie wir sie bereits von anderen, modernen Programmiersprachen her kennen. RPG IV Neueinsteiger, die bereits über Erfahrung in C, COBOL, PL/1 oder in einer anderen Programmiersprache im Free-Format verfügen, werden es leicht haben, Free-Format RPG IV zu erlernen und anzuwenden. RPG IV hat auf diese Weise ein gewisses Maß an Akzeptanz erfahren. Viele der Fix-Format RPG IV „Oldtimer“ sehen geradezu gut aus, wenn sie in Free-Format umgestellt werden.

Für all diejenigen, die bisher noch keine Erfahrung im Bereich der Free-Format-Programmierung haben, ist es zunächst ein bedeutender und nicht ganz unkomplizierter Schritt in Rich-

tung dieser neuen Technik. Ehrlich gesagt, ist dieser Schritt nicht nur unkompliziert, sondern wirklich schwer. Denn das Ablegen lange Jahre erlernter und angewandter Techniken ist kein leichtes Unterfangen. Aber es ist machbar – und wenn Sie ein solcher Anwendungsentwickler sind, der über die Jahre hinweg seine Erfahrungen im Fix-Format-RPG sammeln konnte, der aber noch unbedarft in Sachen Free-Format RPG ist: Dieses Buch wird Ihnen dabei helfen, die ersten Schritte zu tätigen und Ihre Kenntnisse in RPG wesentlich zu erweitern.

Wir werden unsere Tour quer durch das Free-Format RPG IV in diesem Kapitel mit einem Überblick über die Free-Format-Struktur beginnen. Weiterhin werden wir uns einige der wesentlichen Operanten und Features anschauen, die uns in RPG derzeit zur Verfügung stehen.

Der Free-Format Codeblock

Der erste Schritt in Richtung Free-Format RPG besteht darin, dem RPG IV Compiler mitzuteilen, dass Free-Format verwendet werden soll. Wir beginnen einen Block bzw. eine Sektion im Free-Format, indem wir zuerst die Startangabe für das Free-Format in Form von „/Free“, an Stelle 7 des Programmcodes beginnend, vornehmen. Der Rest in dieser Zeile (Positionen 12-80) bleibt leer (Free-Format RPG ist nicht Case-Sensitive – das bedeutet, dass es egal ist, ob die Free-Format-Startanweisung mit /FREE, /Free oder /free angegeben wird – eine Unterscheidung in Groß- bzw. Kleinbuchstaben findet nicht statt).

Der Free-Format RPG Block wird mit der Angabe der Anweisung „/End-free“ abgeschlossen. Sollten Sie die „/End-free“ Anweisung vergessen und es folgt in dem Quellcode keine andere Anweisung mehr, die Fix-Format benötigt, so wird dieses von dem Compiler auch nicht als Fehler gewertet.

Innerhalb der /Free und /End-free Anweisung wird der Free-Format RPG IV-Code platziert. Die einzelnen Anweisungen können an Stelle 8 oder einer anderen, späteren Stelle innerhalb des Free-Format-Blockes beginnen. Die Positionen 1-5 sind nicht für die Angabe von Programmcode verfügbar, sondern dienen lediglich als Informations-Stellen (z. B. für Änderungsmarkierungen). Die Position 6 muss in jedem Falle leer sein, während die Position 7 für zusätzliche Compiler-Direktiven reserviert ist. Die Free-Format Anweisungen enden in jedem Falle mit der Stelle 80 des Quellcodes.

Innerhalb des so definierten Free-Format-Blockes können Sie alle verfügbaren Free-Format Anweisungen sowie alle unterstützten Built-In Functions verwenden. Im Gegensatz zu seinen „älteren Brüdern" im Fix-Format, in denen 112 Anweisungen verwendet werden können, verfügt Free-Format RPG IV derzeit lediglich über 55 Free-Format Anweisungen (Stand V5R3M0). Dabei ist die neue Free-Format-Version keine einfache Abwandlung der Fix-Format Variante, die zudem durch die geringere Anzahl von Standard-Operanten auch noch einen eingeschränkten Funktionsumfang bietet – im Gegenteil! IBM hat eine Vielzahl der „fehlenden" Standard-Operanten durch wesentlich leistungsfähigere Built-In Functions ersetzt. Mit dem Release V5R3M0 werden innerhalb des RPG IV 76 Built-In Functions unterstützt.

Codieren von Free-Format-Anweisungen

Eine Free-Format Zeile beginnt mit einer Free-Format Anweisung, gefolgt von einer oder mehreren Leerstellen, der Anweisung in „Faktor 1", einer oder mehreren Leerstellen und der Angabe des „Faktor 2". Free-Format RPG kennt keine Ergebnisfeld-Anweisung. Stattdessen lassen sich die früheren „Ergebnisfeldfunktionen" mit Hilfe von neuen Free-Format RPG Techniken sehr elegant realisieren.

Eine der Folgen des nicht vorhandenen Ergebnisfeldes ist die fehlende Möglichkeit der Definition von temporären Variablen, wie wir sie seit Jahren als Hilfsfelder in RPG kennen. Aber Dank der ausgereiften Technik des Free-Format RPG entfällt die Notwendigkeit der Definition solcher Hilfsfelder weitestgehend. Die Angabe solcher Hilfsfelder innerhalb von Definitions-Anweisungen wirkt sich zudem positiv auf die Lesbarkeit und die Wartbarkeit solcher Programme aus.

Die letzte Anforderung für eine Free-Format Codezeile innerhalb des RPG IV ist die Verwendung eines abschließenden Semikolons „;“. Diesem Abschlusszeichen können bei Bedarf auch noch erläuternde Hinweise folgen. Kommentare werden nach einem bestimmten Muster angegeben: Der Startpunkt eines Kommentares beginnt stets mit zwei Schrägstrichen (//), auf die der Text bzw. Kommentar folgt. Auf diese Weise können Kommentare auch in einer gesamten Zeile angegeben werden, die sonst der Angabe von Code-Anweisungen vorbehalten ist.

In der Abbildung 2-1 sehen Sie ein Beispiel für die Verwendung eines Free-Format Blockes, der neben Free-Format Anweisungen auch Kommentare enthält.

```
/free
 Miles_per_gallon = Miles / Gallons;
 Eval(h) Pay = Hourly_rate * Hours;
       //  An entire line comment
 Name = %trim(First_n) + ' ' + %trim(Last_n);
 Error_cust_no = *On;    // A short comment on a calculation line
/end-free
```

Abb. 2-1 Beispiel eines Anweisungsblocks im Free-Format

Namensvergabe für Variablen

Es gibt keine Unterschiede bei der Namensvergabe für Variablen im Fix-Format-RPG bzw. im Free-Format RPG. Lediglich bei der Angabe von besonders langen Variablen (Variablenname länger als 14 Zeichen) ist darauf zu achten, dass entweder der Faktor 2 für die erweiterte Angabe des Namens mit verwendet wird bzw. die Angabe des Variablennamens direkt im Free-Format erfolgt. Jeder Variablenname muss mit einem Zeichen beginnen – alle weiteren Stellen können dann wahlweise auch aus Ziffern bestehen. Dabei kann das erste Zeichen jeder der 26 Buchstaben des Alphabetes oder die Sonderzeichen #, $, @ oder _ sein. Die Verwendung der Ziffern 0-9 ist optional möglich – allerdings nicht an der ersten Stelle des Variablennamens. Innerhalb eines Variablennamens sind keine Leerstellen zulässig. Etwa erforderliche Trennungen innerhalb eines Variablennamens lassen sich mit Hilfe eines Unterstriches „_“ realisieren (z. B. Liter_pro_Kilometer).

In der ersten Version des RPG IV, die 1994 erschien, lag die maximale Längenangabe für Variablennamen bei 6 Stellen. Diese Limitierung schloss Referenzierungen für Datenbereiche und deren Indizes ebenfalls mit ein. Zudem war die Länge für Feldnamen limitiert, die, abgestimmt auf die maximale Länge der Feldnamen im DDS, höchstens 10 Stellen lang sein durften. Vor einigen Jahren hat IBM die Längenbeschränkung für Variablennamen weitestgehend aufgehoben und auf eine maximale Länge von 4096 erweitert! Es wird wohl nur wenige Anwendungsentwickler geben, die tatsächlich die gesamte mögliche Länge der Variablen-Namen nutzen werden, die Erweiterung der ursprünglichen Längenbeschränkung und deren sinnvolle Verwendung ist aber ein nicht zu unterschätzender Fortschritt.

Bei der Namensvergabe für Variablen gibt es innerhalb der Free-Format- RPG eine Einschränkung, die es mit Fix-Format nicht gibt. In Free-Format-Rechenbestimmungen dürfen Variablennamen nicht dem Namen eines Operations-Codes übereinstimmen. So sind zum Beispiel die folgenden Bezeichnungen als Variablennamen unzulässig: In, Out, Select, Update usw.. Bei diesen Namen handelt es sich jetzt um reservierte Namen, ähnlich wie es bereits bei anderen Programmiersprachen (z. B. Cobol) auch der Fall ist.

Mathematische Rechenoperationen

Bei der Codierung von mathematischen Rechenoperationen verwenden wir im Free-Format RPG dieselben Operanten, wie wir sie auch aus dem Fix-Format RPG (innerhalb des erweiterten Faktors 2) kennen. Die Standard-Rechenoperationen Addieren, Subtrahieren, Multiplizieren und Dividieren werden in Form von Ausdrücken (+, -, *, /) angegeben. Erstmals in RPG IV hat IBM auch die Möglichkeit geschaffen, exponentielle Operanten (**) einsetzen zu können. Damit lassen sich Exponenten definieren, mit denen dann die Rechenoperationen ausgeführt werden können. Dadurch entfallen bisherige Lösungsansätze, mit denen exponentielle Rechenoperationen nicht direkt innerhalb eines RPG Programms ausgeführt werden konnten, sondern eine Realisierung nur über einen externen C-Programm-Aufruf möglich war. Das Free-Format bietet außerdem mehr Platz für die Definition von komplexen Rechenangaben bzw. Rechenformeln innerhalb einer Anweisungszeile.

Zeichenketten

Die Angabe von Zeichenverkettungen erfolgt in derselben Weise, wie sie auch aus dem Fix-Format RPG bekannt ist (Verwendung des erweiterten Faktors 2). Im Zusammenhang mit der Verarbeitung von Zeichenketten und auch der Umwandlung von numerischen Feldinhalten in Zeichenketten stehen mehrere Built-In Functions zur Verfügung.

Programmier-Stil

Es gibt keine weiteren Einschränkungen bzw. Unterscheidungen zwischen Free-Format RPG und der Fix-Format-Variante. Trotzdem gilt die Empfehlung, sich selbst an gewisse Standards und Regeln zu halten, damit die Lesbarkeit und auch die Wartbarkeit solcher Free-Format Programme gewährleistet ist. Deshalb: Gewöhnen Sie sich von Anfang an einen guten und kontinuierlichen Programmier-Stil an! Ein guter Stil ist zum Beispiel der Beginn der „äußeren" Programmlogik an Stelle 8 des Quellcodes, während die „innere" Logik zwei Stellen weiter rechts beginnt. Dieses Verfahren des „verschobenen Codebeginns" empfehle ich soweit fortzusetzen, bis etwa die Hälfte des Zeilenbereiches erreicht ist. Je tiefer die Programmlogik ineinander verschachtelt ist, desto eher müssen Sie entscheiden, wie Sie diese abbilden – entweder setzen Sie in einem solchen Fall den Code noch weiter rechts beginnend fort oder starten diese Ebenen wieder an Stelle 8 der Sourcecode-Zeile.

Die Abbildung 2-2 zeigt dazu ein Beispiel. In diesem Free-Format RPG-Code werden mehrere Programmebenen abgebildet und durch Einrückungen voneinander getrennt dargestellt.

```
/free
 Dou %eof;
   ReadC SubfileRec;
   If not %eof;
     Fielda = Fieldb;
     If Fieldc <> *zero;
       Error_Msg_1 = *On;
       RI_Fieldc = *On;
       Sflnxtchg = *On;
     Endif;
     Update SubfileRec;
   Endif;
   ReadC SubfileRec;
 Enddo;
/end-free
```

Abb. 2-2 Beispiel für unterschiedliche Programm-Code-Ebenen

Eine Anmerkung zur Großschreibung

Es gibt mit RPG IV keine Regel, nach der Variablennamen, Operationscodes oder Kommentare groß geschrieben werden müssen. Einige Programmierer haben sich selbst einen gewissen „Standard“ angewöhnt: das Großschreiben des ersten Zeichens von Variablennamen bzw. deren Einzelbestandteilen, während alle anderen Buchstaben als Kleinbuchstaben fortgesetzt werden (z. B. SubfileRec). Andere bevorzugen die Version der gesamten Großschreibung von extern beschriebenen Variablennamen. Es bleibt Ihnen überlassen, wann und ob Sie Groß-/Kleinschreibung verwenden. Der Compiler wandelt in jedem Fall für die interne Verarbeitung den von Ihnen eingegebenen Code zunächst in Großbuchstaben um, bevor der Code analysiert wird. Diese Umwandlung wirkt sich allerdings nicht auf den von Ihnen eingegebenen Quellcode aus, sondern ist nur ein „compilerinterner“ Vorgang.

Operations-Codes im Free-Format

Die Tabelle 2-1 enthält die Beschreibung von derzeit 55 Free-Format Operations-Codes, die in dem Release V5R3M0 in RPG IV verfügbar sind. Der Anhang „A“ beschreibt die Verwendung dieser Operations-Codes im Detail. Darüber hinaus werden alle Built-In Functions, die im Fix-Format enthalten sind, ebenfalls im Free-Format unterstützt. Viele dieser Built-In Functions (z. B. %Check, %Lookup, %Scan) können alternativ zu den vergleichbaren Operations-Codes eingesetzt werden. Dabei verfügen einige dieser Built-In Functions über genau denselben Funktionsumfang, wie die vergleichbaren Operations-Codes (z. B. %Check), während andere einen erweiterten Funktionsumfang gegenüber den vergleichbaren Operations-Codes aufweisen (z. B. %Lookup).

Operant	Beschreibung
Acq	Ermitteln einer Programm-Einheit
Begsr	Startpunkt einer Unterroutine
CallP	Aufruf einer Prototyp-Procedur
Chain	Direktzugriff auf einen Datensatz in einer Datei mittels Schlüssel oder relativer Satznummer
Clear	Initialisiert alle einzelnen Elemente einer Datenstruktur, eines Satzformates, einer Feldgruppe oder einer Variablen auf Null (0) oder Leerzeichen (blank)
Close	Schließt eine für die Verarbeitung geöffnete Datei
Commit	Commit-Anweisung für Änderung an Dateien seit der letzten Commit-Rolbk Anweisung
Dealloc	Freigeben von dynamischem Speicher
Delete	Löscht einen Satz in einer Datei
Dou	Do until (logische Gruppierung – Beendigung mit Enddo)
Dow	Do while (logische Gruppierung – Beendigung mit Enddo)

Tabelle 2-1 (Teil 1): Free-Format Operations-Codes

Operant	Beschreibung
Dslpy	Anzeigen einer Nachricht
Dump	Erstellen eines Dumps für ein Programm (Variablen, Satzinhalte etc)
Else	Else-Anweisung (als Teil einer If-Abfrage)
Elseif	Kombination von Else und If-Anweisung (Bestandteil einer If-Gruppe)
Endxx	Beendigung einer End-Gruppe. Der Platzhalter „xx" muss dabei zur Startanweisung der Gruppe passen (z. B. Enddo, Endif, Endmon, Endsl)
Endsr	Ende einer Unterroutine (Subroutine)
Eval	Zuweisen eines Wertes. Wenn eine Zeichenkette angegeben ist, dann wird das Ergebnis linksbündig eingestellt.
EvalR	Zuweisen eines Wertes. Zeichenketten werden rechtsbündig eingestellt
Except	Programmbeschriebene Ausgabe-Anweisung
Exfmt	Schreiben und anschließendes Lesen eines Satzformats (auch als „Execute and Format" bezeichnet)
Exsr	Verzweigen in eine Unterroutine
Feod	Erzwingen des Datenendes
For	Startanweisung einer logischen Anweisungsgruppe. Beeindigung mit „Endfor"
Force	Initiiert eine nächste Leseoperation für eine angegebene Datei
If	Startanweisung einer logischen Anweisungsgruppe. Wird mit „Endif" beendet. Innerhalb der Gruppe sind „else"-Anweisungen möglich.
In	Ruft den Inhalt eines Datenbereiches ab und lädt die angegebene Datenstruktur
Iter	Iterieren bzw. verzweigen zur „Enddo" oder „Endfor" Anweisung

Tabelle 2-1 (Teil 2): Free-Format Operations-Codes

Operant	Beschreibung
Leave	Verlassen einer Subroutine
Monitor	Startanweisung einer Monitor-Gruppe. Beendigung mit „Endmon". Zur Fehlerbehandlung innerhalb eines bestimmten Code-Bereiches
Next	Ausführen der nächsten Input-Aktion von einer angegebenen Einheit
On-Error	Verwendung in einer Monitor-Gruppe zur Fehlerbehandlung
Open	Öffnen einer Datei, für die ein benutzerdefiniertes Öffnen erforderlich ist
Other	Verwendung in einer Select/When Gruppe, wenn keine abgefragte Bedingung zutrifft
Out	Schreiben des Inhaltes einer Datenstruktur in einen Datenbereich
Post	Update der Datei-Informations-Datenstruktur für die angegebene Einheit oder Datei
Read	Liest den nächsten Datensatz. Mit diesem Operant kann ein Dateiname oder Satzformatname angegeben werden
ReadC	Lesen eines veränderten Datensatzes (nur für die Verwendung innerhalb einer Subfile in einer Anzeige-Datei)
ReadE	Read Equal – ein Multifunktions-Operant für das gleichzeitige Vergleichen und Lesen eines Datensatzes mit Hilfe eines vorgegebenen Schlüssels. Wird keine Übereinstimmung gefunden, wird das Kennzeichen „Dateiende" gesetzt.
ReadP	Lesen des vorhergehenden Satzes. Mit diesem Operant kann sowohl ein Dateiname als auch ein Datensatzname angegeben werden.
ReadPE	Lesen des vorhergehenden Satzes. Vergleichbar mit ReadE – allerdings rückwärts lesend)
Rel	Freigeben einer im Programm verwendeten Einheit

Tabelle 2-1 (Teil 3): Free-Format Operations-Codes

Operant	Beschreibung
Reset	Zurücksetzen des Inhaltes eines Feldes auf den Initialisierungs-Wert
Return	Verwendung in zwei Teilbereichen: Am Ende einer Unterroutine oder zur Rückgabe der Steuerung an die Aufrufstelle der Prozedur
Rolbk	Roll Back – zur Verwendung einer Commit-Kontrolle zum Entfernen von Dateiänderungen seit der letzten Commit oder Rolbk Anweisung
Select	Beginn einer logischen Anweisungsgruppe. Erfordert „When"-Anweisung und „Endsl".
Setgt	Setzen des Dateipointers auf den nächsthöheren Wert, bezogen auf den angegebenen Vergleichswert.
Setll	Setzen des Dateipointers auf den Wert, der als Positionswert angegeben wurde. Ist kein Wert innerhalb der Datei vorhanden, dessen Vergleich „Gleich" zur Folge hat, dann wird auf den nächst höheren Dateieintrag positioniert.
SortA	Sortieren eines Datenbereiches.
Test	Gültigkeitstest eines Datumsfeldes, eines Timestamps, eines Zeichenfeldes oder eines numerischen Feldes
Unlock	Freigeben eines Datenbereich-Objektes oder Aufheben einer Satzsperre
Update	Fortschreiben eines Datensatzes, der zuvor mit einer „Read" oder „Chain" Anweisung gelesen wurde
When	Teil einer Select-Anweisungsgruppe.
Write	Schreiben bzw. Hinzufügen eines neuen Datensatzes in eine Datei

Tabelle 2-1 (Teil 4): Free-Format Operations-Codes

Sind es die Operations-Codes, die nicht in dem Free-Format unterstützt werden, die Sie mehr interessieren? Die folgende Tabelle enthält die derzeit als Operations-Anweisungen im Free-Format nicht unterstützten Operanten:

Operant	Operant	Operant
Add	End	Mvr
Call	Goto	Scan
CallB	Lookup	Setoff
CASxx	Move	Seton
Cat	MoveA	Sub
Div	MoveL	Subst
Do	Mult	Tag

Tabelle 2-2: Operations-Codes, die nur im Fix-Format vorhanden sind

Die meisten der im Free-Format fehlenden Operanten haben vergleichbare Ersatz-Anweisungen bzw. Built-In Functions, die alternativ eingesetzt werden können. Damit ist eine einfache Konvertierung auf das Free-Format möglich. Die Anweisungen „Seton“ und „Setoff“ wurden beispielsweise durch die Eval-Operation (z. B. eval *in21=*on) ersetzt. Ein weiteres Beispiel ist der Wegfall der „Mvr“-Anweisung, die neuerdings mit der Built-In Function %Rem realisiert werden kann. Auch die Lookup-Anweisung kann mit Hilfe einer von fünf vorhandenen %Lookup Built-In Functions oder von fünf %Tlookup Functions ersetzt werden. Die Verwendung der unterschiedlichen Versionen der Lookup-Built-In Functions hängt jeweils davon ab, ob Sie zum Beispiel einen Datenbereich oder eine Tabelle durchsuchen wollen.

In dem Anhang C finden Sie eine vollständige Auflistung der im Free-Format nicht unterstützten Operations-Codes und der dazugehörigen Alternativen, die im Free-Format eingesetzt werden können.

Es wäre zu schön, wenn für alle Operanten, die im Free-Format nicht mehr verfügbar sind, entsprechende Alternativen vorhanden wären, deren einfache Konvertierung eine simple Umstellung auf die neue Technologie ermöglichen würde – leider ist das nicht immer der Fall. Einige der Fix-Format-Operanten, nehmen wir als Beispiel die Anweisung „Move", haben keine direkten vergleichbaren bzw. einsetzbaren Free-Format-Alternativen. In dem Kapitel 8 finden Sie für solche speziellen Fälle entsprechende Lösungsvorschläge (wie Sie dann sehen werden: es sind wirklich nur einige wenige Fälle, in denen es keine direkte Free-Format-Alternative gibt!).

Wer weiß – vielleicht bietet IBM uns irgendwann einmal für alle Fix-Format-Operanten auch entsprechende, direkt umsetzbare und anwendbare Alternativen, damit eine vollkommen automatische Umstellung dieser Programme problemlos möglich ist.

Zusätzlich möchte ich Ihnen in diesem Kapitel einige wichtige RPG IV Operanten im Detail vorstellen (und auch einige Built-In Functions) und deren Anwendung anhand von Beispielen verdeutlichen. Sollten Sie bereits den erweiterten Faktor 2 im RPG IV für Rechen- und Anweisungsoperationen verwenden, dann werden Ihnen die Beispiele bekannt und vertraut vorkommen.

Evalulieren (zuweisen eines Wertes)

Der im RPG IV Free-Format am häufigsten benutzte Operant ist ganz genau genommen kein wirklicher Operationscode. Es ist die Definition einer Feldzuweisung ohne die Angabe eines Operations Codes (wie zum Beispiel „Eval“).

Diese besondere Form der Wertzuweisung wird nach einem festen Muster eingegeben bzw. codiert. Dabei wird dem links neben dem Gleichheitszeichen „=“ stehenden Feld der Wert des rechts neben dem Gleichheitszeichen befindlichen Feldes oder der Konstanten zugewiesen Vor dem Zuweisen des Inhaltes wird das aufnehmende Feld initialisiert. Dabei wird auch die Länge des Feldes nach den Standardvorgaben berücksichtigt.

Diese Arbeitsweise ist uns auch von anderen Programmiersprachen, die im Free-Format arbeiten, bekannt. Aber wir kennen ein ähnliches Verfahren bereits von der „klassischen AS/400 Programmierung: Die CL-Anweisung „CHGVAR“ (Change Variable bzw. Ändern einer Variablen) auf dieselbe Art und Weise.

Bei der Durchführung von verschiedenen mathematischen Operationen besteht hin und wieder die Anforderung, das Ergebnis zu runden. Die Rundungsmöglichkeit ist unter anderem auch bei der Verwendung der Eval-Anweisung gegeben. Dabei ist dann direkt hinter der Eingabe „Eval“ ein in Klammern zu setzendes „h“ einzugeben (h) – damit wird dem Compiler mitgeteilt, das eine kaufmännische Rundung für das Ergebnis durchgeführt werden soll.

Ähnlich wie die Erweiterung dieses Operanten mit der Angabe von „(h)“ ist zum Beispiel auch die Erweiterung der Anweisung mit einem in Klammern gesetzten „(r)“ möglich.

In der Abbildung 2-3 finden Sie einige Beispiele für die Wertzuweisungen.

```
D Field30         S                   30
D Field10         S                   10    Inz('ABCDEFGHIJ')
D Field3          S                    3

 /free
  Field30 = Field10;     // Field30 is cleared, and then Field10 is
                         // moved to it, left-justified
  Eval(h) Pay = Hourly_rate * hours;  // The math is performed (with
                         // rounding) and the result assigned to Pay

                // More complex forms
  *In03 = F7 = F3;       // This statement checks to see whether F7
                         // is equal to F3. If yes, *In03 is set to
                         // *On, but if F7 is not equal to F3, *In03
                         // is set to *Off.
  *In21 = *In43 or (Pay > 100);  // This statement sets *In21 to *On
                         // if either *In43 is *On or Pay > 100.
  %subst(Field30:5:4) = 'xyz';  // Positions 5-8 of Field30 are
                         // cleared, and then 'xyz' is moved to 5-7.
  Field3 = *In03 + (Pay > 100) + %eof(FileA);  // Three items are
                         // evaluated as true or false and then
                         // concatenated, with the result of Field3
                         // equal to '000', '001', '010', etc.
 /end-free
```

Abb. 2.3: Beispiele für Zuweisungs-Definitionen