15 章 標準算術演算と 31 桁への拡張

この章では算術演算の３１桁拡張と、標準算術演算について説明します。

15.1 算術演算の３１桁拡張について

COBOL 言語は従来でも 18 桁の誤差無し整数演算を定義していました。これは他言語と比較しても精度の高いものであり、COBOL 言語の特徴の１つです。18 桁でも相当大きい数字だと思いますが、これは兆の上の京という単位で 2 桁を表現出来る事になります。COBOL2002 ではこれが 31 桁に拡張されます。これは京の上の、上の、もう１つ上の「壌（じょう）」を３桁表現出来ます。こんな数字を何に使うのでしょうか？しかし科学技術的な範囲も含めて十分余裕のある桁数を扱える事は良い事ですね。やはり時代の流れが、より高度な情報処理技術を求めている、と言う事が出来るでしょう。

31 桁算術機能は、下記の様な場所で利用出来ます。

（１）数字定数
（２）数字項目、数字編集項目の定義
（３）算術演算およびその中間結果(注)
（４）MOVE 命令他、転記の規則が適用されるところ(注)
（５）浮動小数点データを表現する基数、べき数など

(注)演算の中間結果等については、次項の標準算術演算が関係します。

15.2 標準算術演算とは

標準算術演算とはどの様なものでしょうか。算術演算の性質として、どの処理系でも同じ様に動作すべき範囲と、場合によっては異なる処理系毎に異なった結果を生むかも知れない範囲が有ります。算術演算が異なった結果を生じる、と言うと奇異に聞こえるかも知れませんが、例えば演算途中の中間結果の精度等がこれに当たります。

過去の規格に於いては殆ど共通的な範囲のみを定義し、処理系毎に異なるかも知れない処理範囲については明確な規定は有りませんでした。この為逆に演算結果が曖昧になったり、どの様な結果をもたらすかの定め自体がなかったりと、算術演算についての曖昧性が無視出来ない問題になっていたと言えます。

COBOL2002 では算術演算に関する規定を、次の３種類に分類しました。
（１）共通算術演算（注）
（２）処理系固有算術演算
（３）標準算術演算

つまりどのコンパイラに於いても、この様に動作しなければならない、と定めた範囲が（１）、処理系によって異なるかも知れない扱いの部分が（２）です。ですが（２）の部分についても何も指標が無いと前の規格と大差なくなってしまうので、標準的にはこうある事が望ましい、或いは１つの処理案としてこの様な事が考えられる、という事を纏めたのが（３）の標準算術演算であると言えます。

（注）「共通算術演算」という呼び方はこの章の解説の為に、便宜的にこう呼んだもので有って、規格の中には有りません。

15.3　演算種類毎の規定概要

次に各規定について概観します。

（１）共通算術演算
① 算術演算の中で括弧を、算術演算の優先順を明示・変更する為に使用出来る。括弧内の演算は括弧外の演算に先立って行われる。括弧が入れ子になっている場合、最も内側の組から評価される。
②同一レベルの演算では種類によって、次の優先順位が適用される。

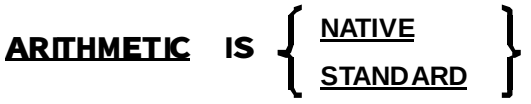
１番目：単項の + と -
２番目：べき乗 **
３番目：乗算 * と除算 /
４番目：加算 + と減算 -

③括弧無しに同じ順位の演算が連続する場合は、左から右へ向かう順序で演算が行われる。括弧を使用するのは、同じ順位の演算が連続している場合の論理的な曖昧さを排除するため、通常の実行順序を変更した順序を指定するため、あるいは通常の順序を強調して明白にする為である。
④算術式中で可能な記号、演算子、作用対象の組合せ（注）
⑤ べき乗の評価規則（注）
（注）詳細は省略

（2）処理系固有算術演算

標準算術演算規則を採用するのか、処理系固有算術演算とするのかの範囲のうち、その処理系で固有の処理に関して実現者は、何が固有で有ってその範囲ではどの様に動作するのかを明示しなければなりません。

見出し部、OPTIONS 段落で次の標記が出来ます。



省略時は NATIVE(処理系固有)です。次の様な演算では上記指定に関わらず常に、処理系固有と見なされます。

①usage of binary-char, binary-short, binary-long, binary-double,float-short, float-long, あるいは float-extended データ項目を含む演算
②べき乗や組み込み関数で近似が使用される場合
③翻訳指令の FLAG-NATIVE-ARITHMETIC が有効な場合

（3）標準算術演算

標準算術演算が有効な場合、下記の演算動作が定義されます。標準算術演算は最も共通的な演算方式で、予測可能、合理的また可搬性のある結果をもたらす事が目的です。ここで可搬性のあるとは、異なる処理系間でも同じ結果を生じる事です。

①標準中間データ項目

標準中間データ項目というのは算術演算の中間結果を格納する為の、数字データ項目であってその内部表現は、処理系作成者が規定する。

②標準中間データ項目が取りうる値は一意なゼロか、抽象的な、符号付き

正規化 10 進浮動小数点形式で、絶対値が 10E-100 以上 10E99 - 10E67 で精度が 10 進 32 桁の範囲である。標準中間データ項目は明示的に指定され無い限り、切り捨てや四捨五入によって 32 桁未満になる事はない。

③ゼロでない正規化された標準中間データ項目は、小数点の左に数字を持たず、小数点の右の桁がゼロ以外の数字である。

④標準中間データ項目は次の場合に 31 桁に四捨五入される。

比較のとき、関数の引数になるとき、及び ROUNDED 指定のない結果データに転記されるとき。

この規則は必要以上に四捨五入が行われる事を排除する。

⑤加算では作用対象の厳密な和を 32 桁へ四捨五入し、正規化し、標準中間データ項目へ格納する。

⑥減算は次の通りとする。

（ 作用対象１ +（- 作用対象２ ））

⑦乗算、除算

加算と同様である。

⑧べき乗

省略。

15.4　標準算術演算の性質

それでは標準算術演算は規格の中で提案されているので、全てこれに従った実現方式が理想という事でしょうか。これはそうでは有りません。標準算術演算はあくまで指針であって、必ず従わなければならない物でも無くまた、従う事が理想という物でも有りません。
例えばある処理系作成者が標準算術演算と異なった、中間データの精度３５桁をサポートしよう、と考えたとします。その精度が必要と思われる根拠があるので有れば、それは立派な態度であり、その点での機能は標準算術演算に従った場合よりも優れているという見方も出来ます。また別の処理系作成者が作成者側の都合から中間データの精度は 31 桁しか確保出来ないとしましょう。その場合でもその様な仕様である事を明示すれば仕様準拠という観点では問題ないのです。

ただどちらの場合でも絶対必要なのは、その様な処理系固有算術演算を採用している事を処理系の仕様書等で明示する事であり、これが守られている限りどちらの処理系も正しく COBOL2002 に対応した物であると言う事が出来ます。

但し標準算術演算の中でも「合理的である」とか「予測可能である」等は重要な標準であるので、守るべきである、とするのが処理系提供者の正しい姿勢と言えるでしょう。