C++のクラスを活用して DSPプログラム開発の効率化を図ろう

三上 直樹

DSPで複雑な演算が含まれるプログラムを開発することが多くなってきました．最近は，浮動小数点演算器をもった高性能なDSPが出回っているため，富豪的なアプローチが可能であれば，そのようなDSPを使って簡単にプログラム開発を行うことができます．

しかし，そのようなことができない場合には，浮動小数点演算器をもたないDSPを使わざるをえないことになります．つまり，固定小数点演算で処理を行うことになるわけですが，スケーリングをつねに頭においてプログラミングを行う必要があり，非常にやっかいです．また，バグが紛れ込む大きな原因にもなりかねません．

このようなときには，C++のクラス(class)と，演算子の多重定義が可能です．したがって，固定小数点演算用に演算子の多重定義を使います．そうすることにより，やっかいなスケーリングの問題を頭におかずにプログラム開発をスムースに進めることが可能です．もちろん，クラスを使わずに関数を定義するだけでもそのようなことはできます．しかし，たとえば，

```
y = (a2*x + a1)*x + a0;
```

と書いた場合と，

```
y = Add(Mpy(Add(Mpy(a2, x), a1), x), a0);
```

と書いた場合では，演算子を使って書いたほうがはるかにわかりやすいのは言うまでもないことでしょう．

ここでは，Texas Instruments社(以下TI社)のTMS320C6000シリーズのDSPを対象にして，固定小数点演算用クラスの例を紹介します．また，それを使った例として，sin関数を固定小数点演算で計算するためのプログラムについて説明します．

なお，ここで紹介するプログラムの開発には，TI社が提供するDSP用プログラム開発環境のCode Composer Studio(以下CCS)のバージョン3.1に付属する最適化C/C++コンパイラを使用しました．

1 固定小数点演算とQフォーマット

● Qフォーマットとは

固定小数点演算を使ってプログラムを作成する際に，小数点の扱い方が大きなポイントになります．TI社のDSP関連の文献では固定小数点データを表現する場合に，Qフォーマットという表現を使っているので，ここでもそれに従うことにします．ある数値を2進数で表現した場合に，小数点以下にnビットを割り当てて表現する形式を「Qnフォーマット」(nの部分には数値が入る)と言います．

たとえば，0.5をQ15フォーマットの16ビット・データ，つまりC++言語のshort型として表現すると，16進数表現で0x4000になります．また，負の数は2の補数表現を行うので，－0.5をQ15フォーマットの16ビット・データとして表現すると，0xC000になります．これらの値の求め方は次のようになります．

$$0.5 \times 2^{15} = 0.5 \times 32768 = 16384 = 0x4000$$
$$-0.5 \times 2^{15} = -0.5 \times 32768 = -16384 = -0x4000$$
$$= 0xC000$$

この計算で，2^{15}(＝32768)を乗算しているのはQ15フォーマットだからです．もしも，Q14フォーマットで表現するのであれば，2^{14}(＝16384)を乗算することになります．

なお，このように変換すると，数値を近似値としてしか表現できない場合も出てくるので注意が必要です．たとえば，0.1をQ15フォーマットに変換しようとすると，次のようになります．

$$0.1 \times 2^{15} = 0.1 \times 32768 = 3276.8 \fallingdotseq 3277 = 0xCCD$$

ただし，これは小数点以下を四捨五入した場合です．したがって，0.1をQ15フォーマットで表現しようとすると近似値になってしまいます．

データxがQ15フォーマットで表現されているものとすると，表現できる数値の範囲は$-1 \leq x < 1$となります．そのため，固定小数点演算でプログラムを作成する際は，Q15フォーマットがよく使われます．

● 固定小数点演算の方法

同じQフォーマットどうしのデータの演算を考えてみましょう．DSPのアプリケーションでは，除算はあまり出てこないので，ここでは加算，減算，乗算について考えてみます．

加算と減算を行う際は，途中で小数点の位置が変わることはないので，とくに何も考える必要はありません．しかし，乗算では小数点の位置が変化するため，考えておく必要があります．

図1にQ15フォーマットのデータをshort型で表した場合

のQ15フォーマットどうしの乗算のようすを示します．そのまま乗算を行うと，この図に示すように，積は小数点以下に30ビット割り当てた形，つまりQ30フォーマットのデータになります．これを，元のデータ形式であるQ15フォーマットに戻すためには，右に15ビットだけシフトします．これが基本的な方法です．

しかし，このままでは下位15ビットを単に切り捨てていることになり，誤差の分布が0を中心に左右対称にはなりません．したがって，誤差の平均値も0にはなりません．

その結果，たとえばディジタル・フィルタの演算をこのような固定小数点演算で行うプログラムを作成した場合，入力信号に直流分が含まれていなくても，出力には直流分が発生する，つまりオフセットが生じることになります．そこで，これを防止するためには，下位15ビットを切り捨てるのではなく，丸め[注1]（10進数の四捨五入に相当）を行わなければなりません[注2]．

図2には，乗算の結果をQ15フォーマットに戻す際に，丸めを行う方法を示します．このように，右シフトを行う前に第14ビット目に1を加えます．こうすると，第14ビット目が0であれば下位15ビットを切り捨てたことになり，第14ビット目が1であれば下位15ビットを切り上げたことになります．したがって，その後に右に15ビットのシフトを行えば，Q30フォーマットのデータの下位15ビットを丸めて，Q15フォーマットに戻したことになります．

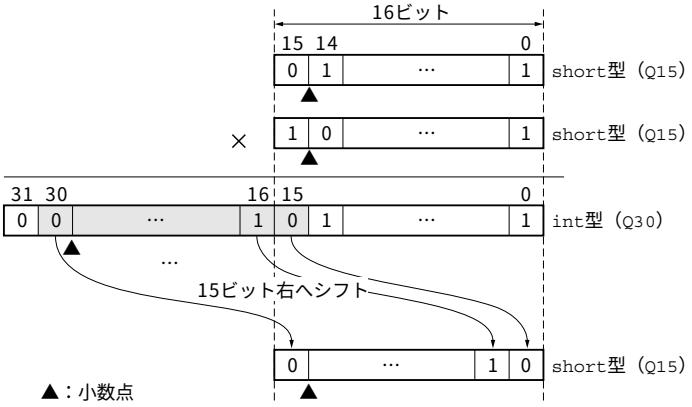

図1　Q15フォーマットどうしの乗算を行い，その積をQ15フォーマットに戻すようす

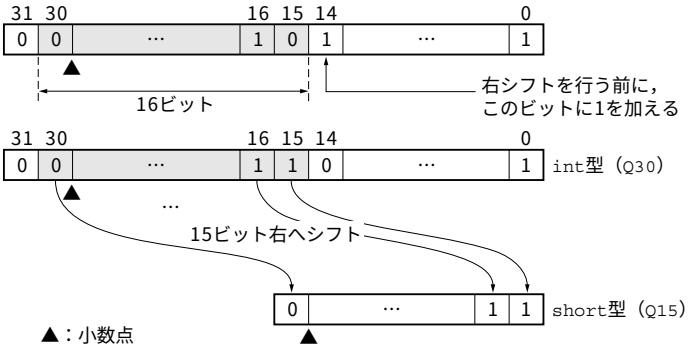

図2　Q30フォーマットのデータに対して丸めを行って，Q15フォーマットに戻すようす

2 固定小数点演算用クラスの作成

ここでは，DSPでの固定小数点演算によるプログラミングを念頭においてクラスを作成します．DSPのアプリケーションでは，四則演算の中で除算はあまり使われないので，加算，減算，乗算のための演算子は定義しますが，除算の演算子は定義しません．算術的な演算子としては，そのほかに単項演算子の“ - ”とシフト演算子を定義します．また，条件文を書く場合も出てくるので，“ == ”や“ > ”などの関係演算子も定義します．さらに，Q15フォーマットとQ30フォーマットの間の変換も必要になります．そのほか，よく使いそうなメンバ関数やフレンド関数を定義します．

ところで，C++言語によるプログラミングでは，エラーはできるだけコンパイル時にチェックするというのが原則の一つです．したがって，ここで作る固定小数点演算用クラスでも，そのような原則に従うように作ります．

リスト1に，今回作成した固定小数点演算用のクラスを示します．この中で，Q15クラスとQ30クラスの二つを定義しています．このリストで，はじめのほうに，

```
class Q30;
```

とあるのは，前方参照のための宣言です．これにより，Q15クラスの定義の中で，Q30クラスを使うことができます．

● Q15クラスとは

▶ フレンド・クラス宣言

最初の，

```
friend class Q30;
```

は，フレンド・クラスの宣言です．この宣言により，Q30クラスはQ15クラスのフレンド・クラスになります．つまり，Q30クラスからQ15クラスの非公開部（private部）のメンバに直接アクセスできるようになります．たとえば，Q30クラスの部分のリストを見ると，x.q15と書かれていますが，このようにQ15クラスの非公開部のデータ・メンバをQ15クラスの外部であるQ30クラスの中からアクセスすることが可能になります．

通常，非公開部にアクセスするためには，そのためのメンバ関数を使います．このQ15クラスでも，Put()とGet()というメンバ関数で，非公開部のデータ・メンバq15にアクセスできるようにしています．しかし，この関数を使ってデータ・メ

注1：「丸め」ということばは，切り捨てや切り上げなどを含めた広い意味で使う場合もあるが，ここでは，10進数における四捨五入に相当する処理（2進数では“ 零捨一入 ”とでも言うべきもの）を丸めと呼ぶことにする．

注2：正確には丸めでも誤差の平均値は0にならない．誤差の平均が0になるのは「偶数丸め(1)」という方法である．これはJISで規定されていることから，「JIS丸め」とも呼ばれている．