

3 J-7 C言語プログラム移植用のC++クラスライブラリの作成[†]

中野和俊

小林純一

原田稔

川村耕基

(沖電気工業株式会社)^{††}

1. はじめに

C言語で記述されたプログラムを異なるアーキテクチャの機種に移植する場合、次のような問題が生じる。ハードウェアでは、エンディアンの違い、アラインメントの違い、ソフトウェアでは、コンパイラ処理系の差(評価順序、シフト演算、符合付きと符合なしの比較)、OSの差がある。(1)

著者らは、上記のような問題をC++のクラスライブラリによってできる限り簡単に解決することを目標としてP S C L (Porting Support Class Library)を試作、評価したのでここに報告する。

2. 概要

P S C Lは、C++言語がC言語の上位互換であることを利用している。C言語で記述されたプログラムを異なる機種に移植する場合に生じる問題を、P S C Lを用いることにより、軽減することを目的としている。P S

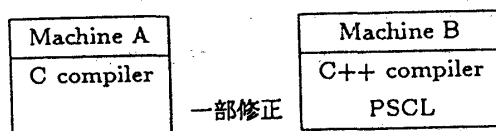
C program $\Rightarrow$ C program

表1 構成図

CLクラスライブラリは移植する元の環境と、移植する先の環境の間で異なる項目、例えばエンディアン、だけをP S C Lを用いて互換性を保つ。2つの環境の間で問題がない項目については、P S C Lを利用しない。従って、速度低下を最小限に抑えることができる。

P S C Lの特徴は以下のとおりである。

- C++言語の抽象データ型機能を用いて、ソースプログラムの修正を減らす。
- 実用的な実行速度

3. エンディアンクラス

ここでは、P S C Lの機能の一例として我々がもっとも大きな問題と考えるエンディアン問題を示し、その問題をP S C Lを使用することにより、解決する方法、評価を述べる。

3.1 エンディアン問題

アーキテクチャが異なる機種間の移植では、エンディアンの相違が問題となる場合がある。例えばC言語のint型がメモリ上に配置される時には次のようになる。

```
int i = 0x12345678;
```

	上位⇄ 下位
Big Endian	12 34 56 78
Little Endian	78 56 34 12

表2 エンディアン

したがってこの変数をint型以外の型で参照した場合、その値はエンディアンに依存するため移植性がない。このようにエンディアンに依存する参照をなくすようにソースプログラムを変更することは、ソースプログラムの大幅な修正を意味する。

3.2 従来手法

この問題点を解決するため、P S S (1)ではC言語の前処理系の指令`#pragma`を使用して、メモリ上の配置を移植元の機種のエンディアンに合わせることを行った。しかし、問題点として次のことが上がっている。

- Cコンパイラに対する改修が大
- 実行速度が低下する。
- コーディング量が減る。
- ソースコードの読解性を確保できる。

3.3 エンディアンクラス

この方法に対し、エンディアンクラスを作成しエンディアンも含めた型を定義することにより、上記の利点を確保しつつ、次のように問題点を克服できる。

- コンパイラの改修の必要はない。ライブラリを作成すれば良い。
- C++のinline展開を利用することにより、

[†] A C++ Class Library to porting C language program by Kazutoshi Nakano, Junichi Kobayashi, Minoru Harada, Kohki Kawamura

^{††} Oki Electric Industry Co. Ltd.

速度低下を最小限に押えられる。

以下にリトルエンディアン形式の `int` 型のクラスの宣言を示す。

リスト 1

```
#define rev(x) ... //xを反転するマクロ
class litint
{
protected:
    int lit;
public:
    litint();
    ~litint();
    litint& operator = (int);
    litint& operator = (litint&);
    operator int() { return rev(lit); }
};
//一部省略
inline litint& litint::operator = (litint &y)
{
    lit = y.lit;    return y;
}
//以下省略
```

3.4 使用例

次に使用例について述べる。

3.2 で示した PSS の方法では次のようになる。

```
#pragma little_endian(i)
int i;
(以下 i のメモリ上の形式はリトルエンディアン形式で格納される。)
```

これに対しエンディアンクラスを使用する場合、次のようになる。

```
litint i;
(以下 i のメモリ上の形式はリトルエンディアン形式で格納される。)
```

`litint` はリトルエンディアンクラスから継承されたクラスで、`int` 型と同じサイズを持ち、メモリ上のデータ配置はリトルエンディアン形式である。ロード/ストアはクラスの中で定義している。

4. 評価

クイックソートプログラムによって評価を行なった。比較する対象は、ソースプログラム中にマクロを書き込む場合と、P S C L を使用する場合である。

結論としては、サイズが若干大きくなるものの、速度

はほぼ同じで、修正は P S C L を使用した方が容易であるため実用的である。

4.1 速度

	最適化 opt なし	最適化 opt あり
C++	22.5	7.3
C	22.4	7.3

表 3 マクロを使用した場合 (単位秒)

	最適化 opt なし	最適化 opt あり
C++	22.5	7.3

表 4 P S C L を使用した場合 (単位秒)

4.2 サイズ

コンパイルしたアセンブルリストの行数で比較した。

	最適化 opt なし	最適化 opt あり
C++	329	236
C	316	224

表 5 マクロを使用した場合 (単位命令数)

	最適化 opt なし	最適化 opt あり
C++	474	356

表 6 P S C L を使用した場合 (単位命令数)

5. おわりに

C 言語プログラム移植用のための C++ クラスライブラリ P S C L を試作した。抽象データ型を導入することにより、C 言語プログラムに対する修正を軽減することができた。

課題としては、(1) C と C++ の非互換の部分があるため (型チェックの厳しさ、ANSI 準拠でないソースプログラムと非互換)、変更箇所が増える場合がある、(2) 既存のライブラリが直接使用できない場合がある、といった点が挙げられている。

今後は、コンパイラの処理系依存項目の差の問題についてクラスライブラリの拡張を行なう予定である。

参考文献

(1) 西風、原田、小林、前田 [PSS: C 言語プログラムの移植支援システム概要]

第 4 4 回全国体会講演論文集 (5) 4 F-7