プログラムの自動変形による C99 規格への移行支援に関する研究

2007MI155 永谷 敏章
指導教員　野呂 昌満

1　はじめに

現在，C 言語の最新の規格は C99 であるが，それ以前の規格に基づいて書かれたソースコードが多く存在する．それらのソースコードを C99 で新たに追加された記述方法を用いることでより簡潔に記述でき，可読性，保守性が向上する．しかし，新しい記述方法を利用してソースコードを手作業で書き換えることは誤りが混入しやすく，また，書き方に制限が加わっている箇所もあり，それらを修正する作業を含めると作業量は非常に多い．

本研究では，新たに追加された記述方法を古い規格で書かれたソースコードに適用し，可読性や保守性を高め，また，規格の違いによるコンパイル時のエラーを解消するする作業を自動化することを目的としている．

2　C99 の規格

C99 での追加点として，予約語の追加，ヘッダファイルの拡張と追加，マクロ定義の追加，標準プラグマ，可変長配列，暗黙の型宣言の削除，前定義識別名，inline 関数，などがある [2][3]．このうち，ヘッダファイルの拡張は主に数学関数であり，元の数式を関数に置き換えるにはその式の意味を知る必要がある．また，前定義識別名や inline 関数は最適化やデバッグに用いるもので，古い規格に対応する書き方がないので，変換の対象に含める必要がない．これらを除いた拡張について以降の節で説明する．

2.1　可変長配列の導入

局所変数として配列を宣言するときにその大きさを変数で指定できる．

2.2　_Bool 型の追加

新たに追加されたヘッダファイル “stdbool.h” に_Bool 型と ture, false がそれぞれ 1 と 0 で定義されている．

2.3　for 文の初期化式の拡張

for 文の初期化式で変数を宣言できる．

2.4　isblank()，iswblank() 関数の導入

改行を除く空白文字を判定する関数である．

2.5　変数の型を省略した場合の処理

C99 以前では特定の箇所で型を省略した場合，暗黙で int 型となるが，C99 では明示的に書く必要がある．

3　プログラムの書き換え方法

ソースコードの書き換えでは，前処理前の状態で変形処理が必要となるので，TEBA[1] を用いる．TEBA は C 言語のソースコードを構文に基づく属性を付与した字句系列に変換する．また，字句系列に対する変換規則を与えることで変形を実現できる．属性には変数，カンマや括弧などの記号，文の始点終点などがあり，“属性 (括弧などの番号) 〈 字句 〉” と表現する．変換規則は “${ 名前:属性 }”で表現し，この属性の並びでパターンを記述する．2.1 節から 2.5 節のうち，for 文と可変長配列の変換は TEBA をそのまま利用しても実現できないので，実現にあたって TEBA の拡張を行う．よって，この 2 つに絞って変換，考察を行う．

3.1　自動化における制約条件

まず，“for 文で使用されているカウンタ変数” と “その変数の宣言文” のように，変数名が同一か確認する必要がある．for 文での制約条件は，変数が使用されている場所が for 文内のみという条件や，2 つ以上の for 文で共通の変数を使用している場合などが挙げられる．また，可変長配列では，malloc() で確保した領域と配列という違いから，書き換えてはいけない場合がある．例えば，malloc による領域の先頭アドレスは他の関数に渡せたり，関数の返り値にできるが，配列ではそれらができない．

3.2　字句解析処理の拡張

これらの制約条件を解決するためには属性の他に識別子名の情報が不可欠である．TEBA では属性情報のみで変換を行うので，属性が変数であればその識別子名に関係なく変換される．この問題を解決するために，変換規則中の “名前” が同一の場合はそれらが指す変数も同一の変数として扱うように TEBA を拡張した．また，制約条件として “特定の場所に特定の変数を含まない” という条件をパターンの中に記述する必要がある．

特定の語を含まないというパターンは正規表現および拡張正規表現でも記述が困難である．そこで, statement 属性の字句に制約条件を指定するオプションを追加した．オプションは_NoInclude_と_NoIncludeFunc_の 2 つを実装し，この後に識別子を記述することで，それぞれ，“その識別子が含まれていない”，“その識別子を実引数に含む関数呼出しがない” ことを表現する．TEBA は識別子とオプションを読み，正規表現と，マッチ後の後方参照を用いて対応する字句列をソースコードの解析結果から抽出する．しかし，オプションの欠点として，オプション付いた字句を規則の最後に記述した場合，指定した変数を含んでいる直前の文までが変換範囲であると認識され，変数を含んでいても変換される．これは変換範囲の終点が，はっきりと示されていないことために起こる問題である．そこで，規則全体をブロックで囲み，終点を “}” と明示する．

また，for 文に関して，ブロック内に for 文が複数あ

り，かつ最初の for 文が適用できず，2 つめの for 文が適用できる場合，2 つめの for 文に適用するために，再度規則を適用しても最初の for 文のみマッチし，変換されない．これを防ぐために，マッチしなかった場合はそのカウンタ変数の属性を変更し，2 回目以降マッチしないようにする．

これらの修正を加えるにあたり，TEBA のプログラム 20 行を修正し，295 行の追加を行った．

4 書き換え規則の適用と評価

4.1 書き換え規則の記述

変数宣言の仕方の違いや，malloc の引数の順序が逆，変数宣言と同時に malloc されている，規則全体をブロックで囲むさいに，はじめの “{” と変数宣言の間に文があるかどうかの違いなど，この規則だけでは補えない部分がある．しかし，字句の並びでパターン化しているので，別の規則として記述し，適用する必要がある．そこで，for 文が 6 通り，可変長配列が 8 通りの規則を実現した．

4.2 適用事例

3.2 節で拡張した TEBA を利用して実際に規則を記述し，書き換えを試みた．

for 文に関する変換について，for 文以外でカウンタ変数が使用されている場合変換されないよう，_NoInclude_ オプションでカウンタ変数を指定した．

可変長配列については，return されているものを除外するために，必ず free() 関数が記述されているものに限定する．また，領域取得から解放までの間に，変数を他の関数に渡している場合，渡した先で free() が記述されている恐れがある．しかし，パターン記述で渡した先の関数内を調べることは困難なので，関数の実引数が先頭アドレスを格納した変数である場合，変換を行わないよう，_NoIncludeFunc_ オプションで条件を指定する．

for 文の変換規則とそれを適用したソースコードの例を図 1，図 2，図 3 に示す．なお，ID_TYPE は変数の型，ID_VAR は変数，EXPR は式，STMT は一つの文またはブロック，STMTS は複数の文を表している．

stmt1，stmt3 にあたる部分は変数 cnt を含まない場合，変換される．また，TEBA 内で再度規則を適用し，for 文の内側の for 文も変換する．

4.3 評価・考察

両規則において，変数宣言より前に別の宣言文がある場合，処理が止まる．別の宣言文を加える前は正しく変換されるので変換規則に問題はないと思われる．変換できない原因として，Perl の置換演算子では対処できない可能性が考えられる．また，可変長配列に関して，途中で例外処理として free() が書かれていた場合，その free() とマッチしてしまい正しく変換が行えなかった．これを防ぐにはその前の if 文の条件式で，それが例外処理だと知る必要があるが，それを確実に知ることは難しいので利用者が判断する必要がある．

```
% before
{
${type:ID_TYPE} ${cnt:ID_VAR};
${stmt1:STMTS_NoInclude_cnt}$;
for(${cnt:ID_VAR} = ${init:EXPR};${cond:EXPR};${succ:EXPR})
${stmt2:STMT}$;
${stmt3:STMTS_NoInclude_cnt}$;
}
% after
{
${stmt1}$;
for(${type} ${cnt} = ${init};${cond};${succ})${stmt2}$;
${stma3}$;
}
% end
```

図 1　for 文に適用した規則の例

```
foo(){
  int i;
  int a;
  bar();
  for(i=0;i<9;i++){
    for(a=0;a<3;a++){
      bar();
    }
    bar();
  }
}
```

図 2　適用前

```
foo(){
  bar();
  for(int i=0;i<9;i++){
  for(int a=0;a<3;a++){
      bar();
    }

    bar();
  }
}
```

図 3　適用後

5 おわりに

本研究では TEBA を利用，拡張し，古い C 言語の規格を C99 で使える記述方法でプログラムをより簡潔にする変換を行った．for 文が 6 通り，可変長配列が 8 通りの規則を定義し，その他_Bool 型が 28 通り，isblank(),iswblank() 関数が 2 通り，暗黙の型宣言に関する変換を 8 通り定義した．for 文，可変長配列は変換するさいに変換部分以外の文を考慮するために上記の問題が起こるが，その他の規則はそれがなく，問題なく変換でき，問題はないと考えている．今後の課題として，変数宣言より前に別の宣言文がある場合に起こる問題を起こさないよう，TEBA を書き換える必要がある．

参考文献

[1] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満，“属性付き字句系列に基づくプログラム書換え支援環境の試作”，ソフトウェアエンジニアリング最前線 (ソフトウェア・エンジニアリング・シンポジウム 2010 予稿集)，pp.119–126，Aug 2010.

[2] 日本工業標準調査会，“プログラミング言語 C”，財団法人 日本規格協会，2003.

[3] seclan，“プログラミング言語 C の新機能”，http://seclan.dll.jp/c99d/