C言語プログラム解析ツール

4S-3

仙田 和彦 ,荻原 啓孝
(株) 東芝 府中工場

1.はじめに

東芝府中工場では、鉄鋼、原子力などの大規模で、複雑な制御システムを多く作成している。この、大規模システムでのプログラム品質を確保するために、TOSBAC-G8000上で動作するチェックツールを報告する。これらのシステムは、複数のタスクで構成しており、タスクレベルで単体試験をしたのちに、組合試験を行う。組合試験では、タスク間で共通的に利用する資源(プリンタ、共用メモリ等)の確保、利用、資源の解放と言う一連の動作を確認する。しかしながら、設計、プログラムの製造誤りにより正しく動作しない場合がある。この種の問題は、複数のタスクが関係するため、この問題のプログラムを特定するために、多くの時間がかかる。

そこで、ソフトウェア作業工程のプログラム製造(対象C言語)が終了した段階でソースプログラムを解析し、これらの資源系の確保/解放の組みの矛盾チェックと、試験実行のための指標を示すこととした。

2.機能

本ツールは、1タスクのC言語ソースプログラムを入力として、静的な解析を行いプログラムレポート、モジュール構成図、共通資源の利用命令の実行順番の確認機能を実現している。

2.1 プログラムレポート機能

ソースプログラムの基本的な品質情報をモジュール別に提供する。

1)ステップ数の表示

バグ発生率がモジュール単位で50ステップを境に急激に増えるためステップ数の表示を行う。

2)コメント数の表示

コメントは、内容の良否判断は行なわないが、ソースプログラムに適量のコメントが記述されているかの判定を行う。

3)goto文数の表示

goto文は、通常は不要であり、使用状況を明確にする。

4)試験時の実行パス数表示

プログラム制御構造文(条件文、繰り返し文、選択文)からすべての実行パス数の表示を行い試験の試験データ量の目安とする。

5)プログラム構造の複雑さの表示

McCabeの循環係数の表示を行い、プログラムの分割、バクの発生率等の調査の基本データとして用いる。

2.2 モジュール構成図

モジュールの呼び合い関係の情報を提供すると共にモジュールを呼ぶためのプログラム構造をあわせて表示することによりモジュール構成をより明確にする。

1)再帰チェック

再帰を明示し不要な再帰を削除させる。

2)階層レベルの表示

モジュールの呼びあい関係のネストの深さの表示を行う。

3)モジュール実行時のプログラム構造表示

任意のモジュールまでの必要なプログラム構造について表示を行う。

4)モジュールへの引き数表示

モジュールへの引き数表示し、引き数誤りをチェックしやすくする。

2.3 共通資源の利用命令の実行順番の確認機能

静的に共通資源の利用命令の実行順番をプログラム構造により良否の判定を実施する。

1)命令の実行順番チェック

命令の実行順についてあらかじめ定義している情報に従いプログラム制御構造(条件文、繰返し文、選択文) を加味してチェックする。これにより、命令の実行順の間違い、命令の抜け、又は、なくてはならない命令が条件文などの条件により実行されない場合のチェックを行う。

2)引き数チェック

引き数の上下限チェック、文字列の比較、ビット情報のチェックを行う。

C language program analyser
Kazuhiko SENDA, Hirotaka OGIWARA
TOSHIBA CORPORATION FUCHU WORKS

2.3.1 解析例

ファイル処理のプログラムmain.c、sub1.cにより、実行順番の確認例とモジュール構成図を示す。

file main.c

```
line no.  1   char r1[32],r2[32] ;
          2   main()
          3   {
          4     int s,i,flag,data;
          5     flag = 0;
          6     dms$fopen(r1,"update","no");
          7     while (flag == 0) {
          8       dms$fget(r1,s) ;
          9       for (;s>0;s--) {
         10           dms$fget(r1,data) ;
         11           if (data == 0) {
         12              dms$fopen(r2,"input","auto") ;
         13              flag = 1;
         14              i=s;
         15           }
         16       }
         17       sub1(i) ;
         18       dms$fstore(r1,9999) ;
         19       fms$fclose(r2) ;
         20    }
         21    dms$fclose(r1) ;
         22  }
```

file sub1.c

```
line no.  1   char r1[32],r2[32] ;
          2   sub1(i)
          3     int  i ;
          4   {
          5     int s,data;
          6     for (;i>0;i--) {
          7       dms$fget(r2,data) ;
          8     }
          9   }
```

mainプログラムの6,8,10,12,18,19,21行、sub1の7行が共通資源の利用である。"r1"と"r2"は、資源に付けた名前で別の資源を示す。ここで、r2の資源は、mainの12行で資源の確保(ファイルのオープン)、mainの19行で資源の解放(クローズ)及び、sub1の7行で利用(読込み)を実施している。

しかしながら、資源の確保は、for文の中のif文で実施しており、この文は実行されないことがある。この様なときに、下記のように警告を発生する。

```
*** open error of DMS file ***
 resource no.              2
 close   pass{file no.(#line no.)}  #19
 not open pass          else  #9
*** open error of DMS file ***
 resource no.              2
 close   pass{file no.(#line no.)}  #19
 not open pass          else  #11
```

モジュール構成図

```
input file name :   :  main.c
input file name :  1:  sub1.c

file line no.  module list
name

           3  main( )
           6    |- dms$fopen(r1,"update","no")
           7    |- while ( flag==0 )
           8    |     |- dms$fget(r1,s)
           9    |     |- for ( ;s>0;s-- )
          10    |     |     |- dms$fget(r1,data)
          11    |     |     |- if ( data==0 )
          12    |     |     |     |- dms$fopen(r2,"input","auto")
          17    |     |- sub1(i)
   1      6    |     |     |- for ( ;i>0;i-- )
   1      7    |     |     |     |- dms$fget(r2,data)
          18    |     |- dms$fstore(r1,9999)
          19    |     |- fms$fclose(r2)
          21    |- dms$fclose(r1)
```

4.まとめ

1) ステップ数、コメントの量、試験時の実行パス数、プログラムの複雑度の表示により、試験結果からプログラムスタイルの見直しのためのデータが得られるようになった。
2) モジュール構成図にプログラム制御構造(条件文、繰返し文、選択文)を追加することにより任意のモジュールに達するまでの条件式が明確になり試験時の支援情報となった。
3) 実行命令のチェックにより、従来の手作業から機械化により効率のアップと不具合の早期発見により後戻り作業が少なくなった。
   なお、実行命令の順番について警告が出た時の最終判断は、設計者が行っている。

5.課題

1) 品質管理の基本データの採取までは、実施したが、試験結果との関係付けなどは分析できていないので、さらに、品質データとバグ数との関連分析を実施していく。
2) 共通資源を資源名で対応付けているが、1部のプログラムで同じ資源を異なる資源名で使用しているものもあり、この場合に対応が付かないために警告が多くでた。今後は、この様な場合も対応付けできるような対応を検討する。