1 SUMMARY

This subroutine is a Fortran profiler. It **inserts extra statements into a Fortran source deck to count the number of executions of each source statement and accumulate the time spent in each subroutine** when entered by each chain of calls. It is assumed that the source is valid standard Fortran 77. The case is treated as insignificant in language keywords.

ATTRIBUTES — Version: 1.0.0. **Remark:** This is a rewritten version of OE02, allowing it to handle Fortran 77. **Types:** OE12A. **Calls:** OE12 calls OE16 and the modified source calls OE17. **Original date:** July 1986. **Origin:** J.K.Reid, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Argument List

```
CALL OE12A(NIN,NOUT,LABEL,MAXL,NAME,NCOUNT,MAXN,ISTART,LP)
```

NIN is an INTEGER variable which must be set by the user to the unit number for input of the Fortran data. This argument is not altered by the subroutine.

NOUT is an INTEGER variable which must be set by the user to the unit number for output of the Fortran code with additional statements. This argument is not altered by the subroutine.

LABEL is an INTEGER workspace array of length MAXL.

MAXL is an INTEGER variable which must be set by the user to the length of the array LABEL. It indicates the limit on the number of extra statements that can be inserted (two-thirds of the number of source statements is usually adequate). This argument is not altered by the subroutine.

NAME is a CHARACTER*6 workspace array of length MAXN.

NCOUNT is an INTEGER workspace array of length MAXN.

MAXN is an INTEGER variable which must be set by the user to the length of arrays NAME and NCOUNT. It indicates the limit on the number of subprograms and entries that may be treated. This argument is not altered by the subroutine.

ISTART is an INTEGER which the user sets to zero unless the program being processed is large (see §2.5). This argument is not altered by the subroutine.

LP is an INTEGER which the user must set to the unit number of the device on which output is to be printed when the modified program is executed (normally 6 for the line printer). This argument is not altered by the subroutine.

2.2 Common

The subroutine contains the following common block

```
COMMON/OE12Y/LW
```

LW is an INTEGER variable which may be set by the user to the limit on the number of chains of calls. Its default value is 1000, so should more be needed the common statement must be inserted in the main program, and the required value assigned to LW.

If LW is reset, the processed program will not conform to the standard unless the OE17 common arrays are altered to have length LW.

2.3 Execution overheads and control statements

If all the user's subprograms are long, then typically the transformed program runs 20-30 percent slower than the original. The overheads may, however, be much greater if there is heavy execution of short subprograms because of calls to the timer. Special comments of the form `C.ON.` and `C.OFF.` (in columns 1 to 5 and 1 to 6) control whether time calls are inserted. If the last one before a particular subprogram is `C.OFF.` then no timer calls are made in that subprogram. These special comments may be inserted anywhere in the source.

2.4 Resetting and printing the counts and times

The counts and accumulated times may be reset to zero during execution of the processed Fortran program by the statement

```
CALL OE12B
```

being inserted in the user's Fortran program. Such a statement is automatically inserted at the beginning of the main program. The counts may be printed by inserting the statement

```
CALL OE12C(LP)
```

where `LP` is an `INTEGER` specifying the printing unit. Such a statement is automatically inserted before each `STOP` or `CALL EXIT` statement.

A program can be treated without any change to the user's source, but these control statements allow the user to make selective counts.

2.5 Development of large programs

During development of a program there are likely to be a large number of runs, each requiring the processing of the source and the compilation of the processed code. For short programs it is satisfactory to process the whole program at each run, but for long programs it is better to keep a load module and recompile only the new and changed subprograms. All such programs must be presented at each later run.

On first processing the program `OE12A` should be called with `ISTART=0`. Later runs must be with `ISTART` one more than the subscript *i* of the last extra statement

```
OE12E(i) = OE12E(i) + 1
```

inserted in the original run.

3 GENERAL INFORMATION

Workspace: Arrays `LABEL`, `NAME` and `NCOUNT` are used for workspace.

Use of common:

(i) `OE12A` uses the common area `OE12Y` (see §2.2);

(ii) the user's modified code contains references to common areas `OE17D/E/F/G/H/I` (see `OE17A` write-up).

Other routines called directly:

(i) `OE12A` calls `OE16A`;

(ii) The user's modified source calls `OE17A`.

Input/output:

(i) `OE12A` reads from unit `NIN` and outputs on unit `NOUT`;

(ii) The user's modified code outputs on unit `LP`.

Restrictions: MAXL limits the number of inserted statements. MAXN limits the number of entries that can be monitored.

System dependence: Through OE17A the user's modified code depends on the timer ZA02AS.

4 METHOD

OE12A reads the user's source in a single pass (so that it cannot add extra statement labels). It hands the source statement by statement to OE16A. This subroutine parses each statement sufficiently to permit OE12A to recognise all branch points and the beginnings and ends of subprograms. It inserts statements in the user's code, displacing them to the right in order not to interfere with readability. At the head of each subprogram it inserts

```
COMMON/OE12D/OE12E(maxl)
COMMON/OE12X/OE12N
INTEGER OE12E,OE12J,OE12N
LOGICAL OE12L
```

where *maxl* is the value of the variable MAXL. At each branch point (subprogram head, label, DO, DO terminator, block IF control statements) it inserts

```
OE12E(i) = OE12E(i) + 1
```

so that the array OE12E accumulates the number of times each block of code is executed. Logical IFs are treated by replacing

```
IF(logical expression) statement
```

by

```
OE12L=logical expression
IF(OE12L)OE12E(i)=OE12E(i)+1
IF(OE12L) statement
```

At the head of the main program the statement

```
CALL OE12B
```

is inserted to perform initializations and before any STOP or CALL EXIT statement OE12A inserts

```
CALL OE12C(lp)
```

to print the accumulated counts and time.

Unless timing has been inhibited with the special comment C.OFF. (see §2.3) the statement

```
DATA OE12J/2/
```

is inserted at the subroutine head, the statement

```
CALL OE17A(OE12J,name)
```

is inserted after each entry point, where *name* is the name of the program, subprogram or entry, and the statement

```
CALL OE17B(OE12J)
```

is inserted before each RETURN. The variable OE12J is reset to 1 by OE17A and to 2 by OE17B and no action is taken by OE17A on a call with OE12J=1; this is necessary because entry points may be passed during normal execution.

An END statement not preceded by a STOP in a main program, or by a RETURN in a subroutine or function subprogram, will normally have a STOP or RETURN inserted, with its associated CALL. However, if the END statement is preceded by an unconditional or assigned GO TO or by an arithmetic if, and the END statement is unlabelled, no such insertions are made, since they would never be executed. A RETURN statement in a main program will be treated like a STOP.

Blocks of more than 25 comment cards will be reduced to 25 lines by writing only the first 24 and the last one in the processed code. Comment cards within a continued statement will be moved to the front of the statement.

The OE17 group of subroutines accumulate timing and calling sequence information in the common areas OE17E/F/G/H/I.

Once processing of the user's source is complete, subroutines for initializing and printing the accumulated counts and times are generated and output. When first processing a program (ISTART=0), the following are generated.

(i) `SUBROUTINE OE12B`

This initializes the counts and times, declares the OE17 common blocks in case LW (see §2.2) has been set to give different arrays sizes. It calls OE12P to find how many counts have been inserted.

(ii)
```
SUBROUTINE OE12P(NCNT)
INTEGER NCNT
```

This simply returns the number of counting statements.

(iii)
```
SUBROUTINE OE12C(LP)
INTEGER LP
```

This prints the counts and times on unit LP. It contains the names of the user's entries and the number of associated counts in DATA statements. Where a subprogram has several entries all the counts are regarded as belonging to the last entry, so that timing information is printed for all entries ahead of the associated counts. It calls OE12P to find the number of counts. It also calls OE12M to obtain additional subroutine names and numbers of counting statements.

(iv)
```
SUBROUTINE OE12M (SUB,NUM,LSUB,NSUB)
CHARACTER*6 SUB(LSUB)
INTEGER NUM(LSUB),LSUB,NSUB
```

On this occasion this executes an immediate RETURN.

When inputting new and changed subprograms for an existing load module (ISTART>0) new versions of OE12P and OE12M are produced to supersede the old ones. The new OE12P passes the revised number of counting statements and the new OE12M adds the names and numbers of count statements for the new entries into SUBC and NUMC and increases NSUB to the new total number of entries processed.

5 EXAMPLE OF USE

A source program may be processed and output into a data set or partitioned data set member, so that the code can then be compiled and executed in the normal manner. This may be done as follows. Assume that unit 8 contains the source fortran, unit 6 has been allocated to the printer and unit 9 will contain the processed source fortran with additional statements.

```
INTEGER LABEL(2000),NCOUNT(100)
CHARACTER*6 NAME(100)
CALL OE12A(8,9,LABEL,2000,NAME,NCOUNT,100,0,6)
STOP
END
```