

Modular Programming in MPE

Ingenieurbüro Jörg Grössler

IIG, Gbgh, Berlin

MODULAR PROGRAMMING

- There is no final definition yet
- A module can be embedded into any environment knowing its interface but not the algorithm used.

example:

$\sin(x)$

the user must know:

- x must be of type "REAL"
- $\sin(x)$ will be of type "REAL"
- $\sin(3.1415) = 0$
- $1.2E-50 < x < 4.5E+55$
- what happens in case of error

the user must not know:

- the method how $\sin(x)$ is calculated

SOME MORE ASPECTS

- A module can be constructed without knowing the environment it will be used in

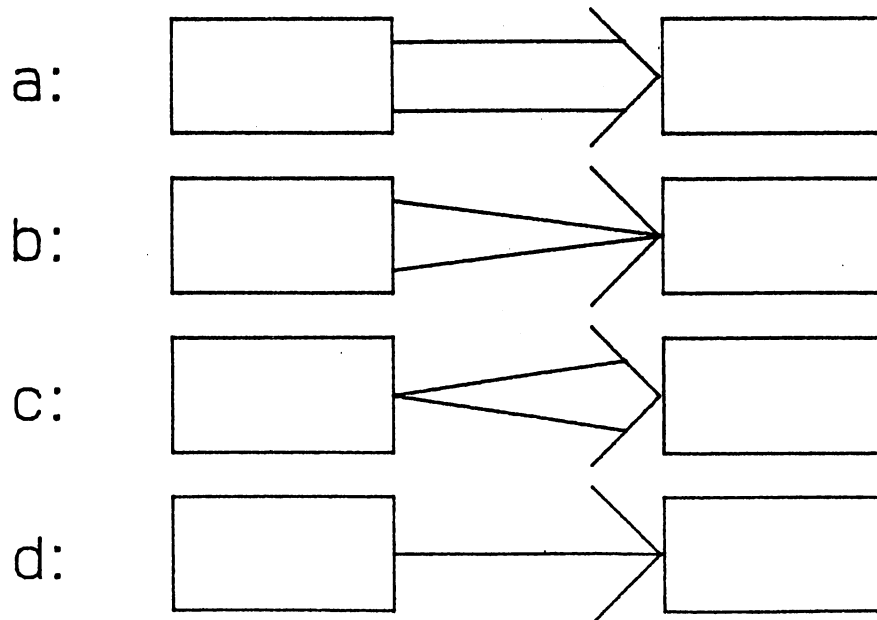
- The module interfaces should be as simple as possible

WHAT MODULES CAN OFFER

- Procedures
e.g.: $\sin(x)$
- Data
e.g.: INTEGER ARRAY A
- Files
e.g.: Data-base
- Any mixture of the three above

MODULE INTERFACES

- Information flow between modules
- Described by:
 - The type of information (data, procedure, file)
 - The access rights for each communication direction:



Examples for Module Interfaces

```

a:  BEGIN
      INTEGER I;
      ...
      PROCEDURE P1;
      BEGIN
        I:=0;
        WHILE (I:=I+1) < 10 DO
          BEGIN .... END;
        END;
      ...
      I: *EQL0;
      WHILE (I:=I+1) < 10 DO
        BEGIN
          ...
          P1;
        END;
      ...
    END;

c:  SUBROUTINE SUB
      ...
      INVAL=ITEMP (10)
      ...
    END

```

MODULE REQUIREMENTS

- Control of information flow (specification of imported and exported objects)
- Check of interfaces (some checking done by SEGMENTER, but not for all types)
- Hidden information (to keep information within the module — problems with stack-structure, file-access)
- More possibilities to restrict access on data, procedures and files
- Comfortable to handle (library-problem)

Example: Own Data in SL-Routines

PROBLEM: The principle of hidden information requires that local data is not deleted between two procedure calls. This causes problems when procedure has to be put into a SL.

WHAT WE WANT: A module which stores local data into an extra data segment before exit and refreshes the data after call.

SPECIFICATION FOR MODULE "OWN DATA"

```

PROCEDURE INITDATA (BUFFER, LENGTH);
INTEGER ARRAY BUFFER;
VALUE LENGTH; INTEGER LENGTH;
OPTION EXTERNAL;

BEGIN
  IF 'first time used'
  THEN 'initialize BUFFER with 0'
  ELSE 'refresh BUFFER with data
        stored in data segment';
END;

```

```

PROCEDURE UPDATEDATA (BUFFER, LENGTH);
INTEGER ARRAY BUFFER;
VALUE LENGTH; INTEGER LENGTH;
OPTION EXTERNAL;

BEGIN
  'copy contents of BUFFER into
  data segment';
END

```

Solution No. 1

```

PROCEDURE INITDATA (BUFFER, LENGTH);
INTEGER ARRAY BUFFER;
VALUE LENGTH; INTEGER LENGTH;

BEGIN
  'allocate data segment';
  IF 'data segment already exists'
  THEN 'copy contents into BUFFER'
  ELSE 'initialize BUFFER with 0';
END;

```

```

PROCEDURE UPDATEDATA (BUFFER, LENGTH);
INTEGER ARRAY BUFFER;
VALUE LENGTH; INTEGER LENGTH;

BEGIN
  'allocate data segment';
  'copy contents of BUFFER into
  data segment';
END;

```

But

- Extra data segment has to be "global."
- Therefore:
- Other users of module "OWN DATA" will use the same data segment
 - Data segment is not automatically deallocated when program terminates. So no initialization will happen after the module has been used once.

Solution No. 2

```

PROCEDURE INITIALIZEDATE;
OPTION PRELUDE;

BEGIN
  'allocate extra data segment';
  'mark user within data segment';
  'initialize info part';
END;

```

```

PROCEDURE INITDATA (BUFFER, LENGTH);
  INTEGER ARRAY BUFFER;
  VALUE LENGTH; INTEGER LENGTH;

  BEGIN
    'allocate extra data segment';
    IF 'used first time (info part)'
      THEN
        BEGIN
          'initialize BUFFER with 0';
          'change info part';
        END
      ELSE 'copy contents into BUFFER';
  END;

```

```

PROCEDURE UPDATEDATA (BUFFER, LENGTH);
  INTEGER ARRAY BUFFER;
  VALUE LENGTH; INTEGER LENGTH;

  BEGIN
    'allocate extra data segment';
    'copy contents of BUFFER into
    data segment';
  END;

```

```

PROCEDURE FREEDATA;
  OPTION POSTLUDE;

  BEGIN
    'allocate extra data segment';
    'delete module user from info
    part';
    'free extra data segment';
  END;

```

