Problemi con la programmazione procedurale

Luca Lista

INFN, Sezione di Napoli

Programmazione procedurale

- Uno dei principali problemi del software è la sua evoluzione e la sua manutenzione
  - specialmente in grossi progetti
- Esempi con linguaggi procedurali (es.: C, Pascal, Fortran):
  - Cosa succede quando il codice viene modificato
  - Dipendenze all'interno del codice

Un esempio semplice

- Un esempio “elegante”: copiare una sequenza di caratteri dalla tastiera alla stampante

```
      SUBROUTINE COPY
        EXTERNAL READKB
        LOGICAL READKB
        EXTERNAL WRTPRN
        CHARACTER C

        DO WHILE (READKB(C))
          CALL WRTPRN(C)
        ENDDO
      RETURN
```

- Questo codice dovrà prima o poi essere modificato (aggiornamenti, estensioni, richieste dagli utenti, ecc.)

Modifiche dei requisiti

- Una modifica un po' meno elegante: scrivere anche su un file

```
      SUBROUTINE COPY(FLAG)
        EXTERNAL READKB
        LOGICAL READKB
        EXTERNAL WRTPRN, WRTFL
        INTEGER FLAG
        CHARACTER C

        DO WHILE (READKB(C))
          IF (FLAG .EQ. 1)
            CALL WRTPRN(C)
          ELSE
            CALL WRTFL(C)
          ENDIF
        ENDDO
      RETURN
```

Evoluzione incontrollata

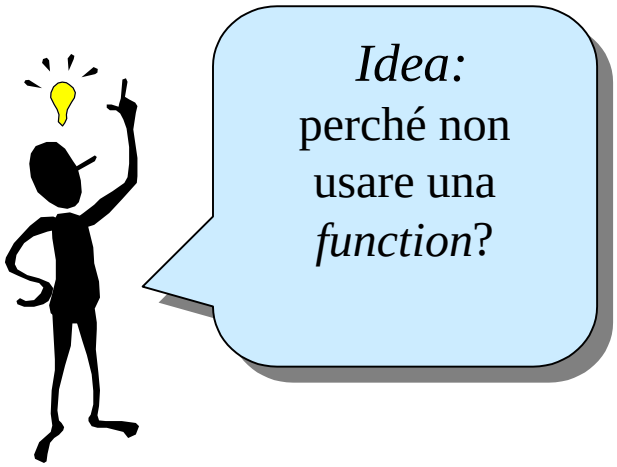
- Un’altra modifica per niente elegante: leggere anche de un file

```
          SUBROUTINE COPY(FLAG1, FLAG2)
            EXTERNAL READKB, READFL
            EXTERNAL WRTPRN, WRTFL
           LOGICAL READKB, READFL
            INTEGER FLAG1, FLAG2
           LOGICAL CONT
            CHARACTER C

10          IF (FLAG1 .EQ. 1) THEN
              CONT = READKB(C)
            ELSE
              CONT = READFL(C)
            ENDIF
            IF (CONT) THEN
              IF (FLAG2 .EQ. 1) THEN
               CALL WRTPRN(C)
              ELSE
                CALL WRTFL(C)
              ENDIF
              GOTO 10
            ENDIF
          RETURN
```

Descrizione dei dati

- Esempio: calcolo degli angoli tra 4 vettori




```
      COMMON /MYDATA/ V1(3), V2(3),
     +               V3(3), V4(3)
      REAL V1(3), V2(3), V3(3), V4(3)

      COSTHETA12 = (V1(1)*V2(1) + V1(2)*V2(2) +
     +              V1(3)*V2(3))/...
      COSTHETA13 = (V1(1)*V3(1) + V1(2)*V3(2) +
     +              V1(3)*V3(3))/...
      COSTHETA14 = (V1(1)*V4(1) + V1(2)*V4(2) +
     +              V1(3)*V4(3))/...
```

```
      FUNCTION COSTHETA(V1, V2)
        REAL V1(4), V2(4)
        COSTHETA = (V1(1)*V2(1) + V1(2)*V2(2) +
     +              V1(3)*V2(3))/...
      END
```

```
      COMMON /MYDATA/ V1(3), V2(3),
     +               V3(3), V4(3)
      REAL V1(3), V2(3), V3(3), V4(3)

      COSTHETA12 = COSTHETA(V1, V2)
      COSTHETA13 = COSTHETA(V1, V3)
      COSTHETA14 = COSTHETA(V1, V4)
```

Evoluzione del codice

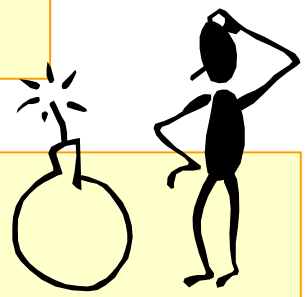
- Se cambia il formato del common block?

```
      COMMON /MYDATA/ R(4),
     +                THETA(4), PHI(4)
```

- Bisogna cambiare la funzione (gli argomenti sono diversi)

```
       FUNCTION COSTHETA1(THETA1, THETA2,
      +                   PHI1,   PHI2)
         COSTHETA1 = SIN(THETA1)*SIN(THETA2) *
      +     COS(PHI1-PHI2) + COS(THETA1)*COS(THETA2)
        END
```

- …e il codice!



```
       COMMON /MYDATA/ R(4),
      +                THETA(4), PHI(4)

       COSTHETA12 = COSTHETA1(THETA(1),THETA(2),
      +             PHI(1), PHI(2))
       COSTHETA13 = COSTHETA1(THETA(1),THETA(3),
      +             PHI(1), PHI(3))
       COSTHETA14 = COSTHETA1(THETA(1),THETA(4),
      +             PHI(1), PHI(4))
```

Il concetto di dipendenza

- Nell'esempio precedente il codice di analisi dati ("*alto livello*") dipende dai dettagli della struttura dati ("*basso livello*").

```
    FUNCTION COSTHETA(V1, V2)
      REAL V1(3), V2(3)
      COSTHETA = (V1(1)*V2(1) + V1(2)*V2(2) +
     +            V1(3)*V2(3))/...
    END
```

COSTHETA dipende dalla struttura dei dati **V1** e **V2**

```
    COMMON /MYDATA/ V1(3), V2(3),
     +              V3(3), V4(3)

    COSTHETA12 = COSTHETA(V1, V2)
    COSTHETA13 = COSTHETA(V1, V3)
    COSTHETA14 = COSTHETA(V1, V4)
```

Il codice di analisi dipende dalla struttura del common block **MYDATA**

Object Oriented / C++

- Riduce la dipendenza del codice di alto livello dalla rappresentazione dei dati
  Permette il riutilizzo del codice di alto livello
- Nasconde i dettagli di implementazione
  Supporta tipi di dati astratti

  (*vedere seguito* → ...)