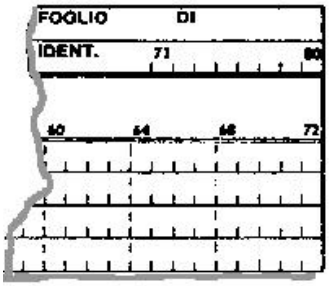
COmmon Business Oriented Language

Creato sul finire degli Anni ’50 su ordine del Department of Defense degli Stati Uniti per poter soddisfare alle seguenti specifiche:

- linguaggio indipendente dalla marca o dal modello del computer; aperto e di forma discorsiva
- linguaggio estensibile, adattabile cioè alle macchine di successiva generazione
- linguaggio facile da apprendere, di approccio agevole per i principianti
- i programmi scritti nel nuovo linguaggio avrebbero dovuto essere autoesplicativi, "leggibili" sia da dirigenti che da profani

Il Foglio di Programmazione COBOL



Colonne 1-6

Possono contenere un numero composto nella seguente maniera: le prime tre cifre individuano il numero del foglio, le rimanenti il numero della riga all'interno del foglio
Può contenere solo alcuni simboli particolari:

***** identifica la riga come riga di commento
- la riga è la continuazione della precedente
/ viene effettuato un avanzamento pagina

Colonna 7

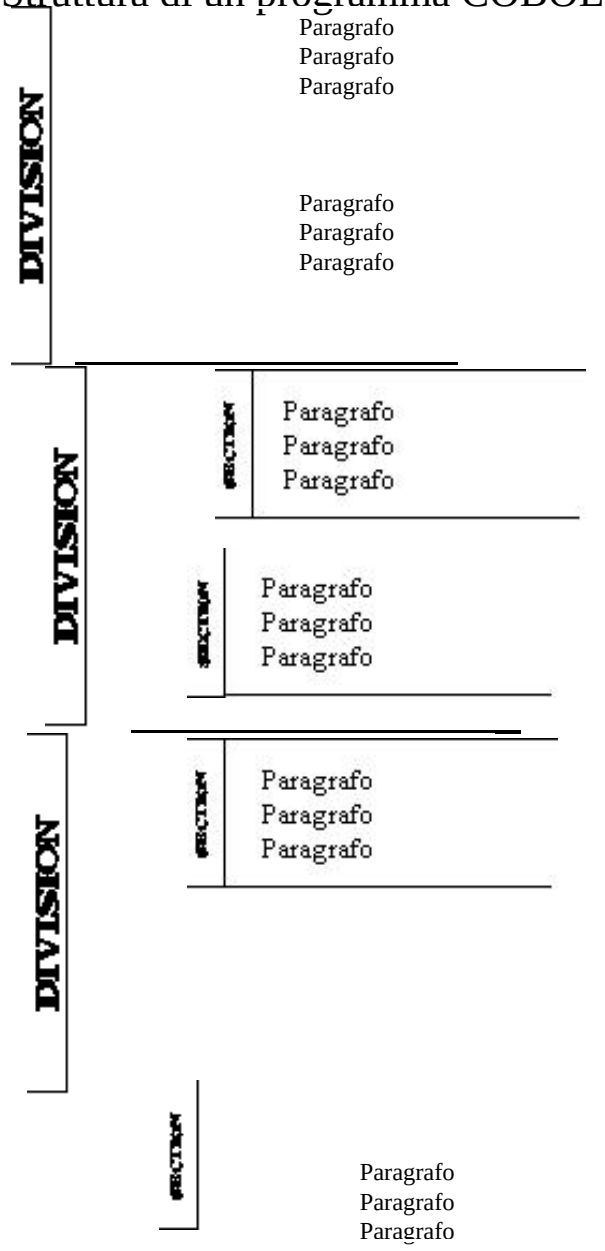
Colonne 8-11

Identificata dalla lettera **A** a colonna 8, questa zona è detta **Area A** ed è riservata ai nomi di Divisioni, Sezioni e Paragrafi (parti del programma COBOL)

Colonne 12-72

Identificata dalla lettera **B** a colonna 12, questa zona è detta **Area B** ed è riservata alle istruzioni del programma COBOL

Struttura di un programma COBOL - 1



Ogni programma COBOL è strutturato in divisioni (**DIVISION**), sezioni (**SECTION**), paragrafi e periodi, organizzati gerarchicamente.
Un **paragrafo** è costituito da più periodi ed è individuato da un nome scelto dal programmatore
Un **periodo** è un insieme di istruzioni (al limite anche una sola) che termina con un punto

Struttura di un programma COBOL - 2

Un programma COBOL è costituito **sempre** da quattro divisioni disposte in sequenza. Tale sequenza consente al Compilatore di acquisire in successione logica tutte le informazioni indispensabili per la corretta traduzione del programma stesso

IDENTIFICATION DIVISION

Consente di dichiarare al Compilatore il nome del programma.
Si possono inserire altre informazioni facoltative di documentazione.
È l'unica divisione composta esclusivamente da paragrafi

ENVIRONMENT DIVISION

Consente di dichiarare al Compilatore l'ambiente in cui il programma dovrà operare, descrivendo le associazioni fra i files usati nel programma ed i mezzi esterni su cui questi files sono definiti

DATA DIVISION

Consente di dichiarare al Compilatore la struttura dati richiesta dal programma.
I dati sono divisi in: a) dati di ingresso uscita, b) dati ricavati da elaborazioni intermedie, c) dati per lo scambio con eventuali subroutines esterne

PROCEDURE DIVISION

È la divisione nella quale sono organizzate tutte le istruzioni COBOL.
In genere consiste in un insieme di paragrafi e sezioni in cui sono disposti **raggruppamenti logici** di istruzioni

IDENTIFICATION DIVISION

Il suo nome, così come quello dei suoi paragrafi si scrive a partire dal **margine A** (colonna 8).
L'unico paragrafo indispensabile è quello che contiene il nome del programma.

FORMATO

IDENTIFICATION DIVISION.
PROGRAM-ID. nome-programma.

ENVIRONMENT DIVISION

Il suo nome, così come quello delle sue sezioni e paragrafi si scrive a partire dal **margine A** (colonna 8).
Nella prima sezione (composta da due paragrafi) viene specificato l'hardware sul quale è compilato ed eseguito il programma.

FORMATO

IDENTIFICATION DIVISION.
PROGRAM-ID. nome-programma.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. nome-computer**.**
OBJECT-COMPUTER. nome-computer**.**

Questa sarà, praticamente, l'intestazione di qualunque programma COBOL

DATA DIVISION - 1

I dati sono descritti utilizzando i **numeri di livello** per evidenziare il loro livello gerarchico e le **PICTURES** per rappresentare in forma simbolica le caratteristiche dei dati stessi.

PICTURES

9 indica un carattere numerico
X indica un carattere alfanumerico
A indica un carattere alfabetico

numeri di livello

01	è sempre associato alla struttura o al record e individua gli stessi nel loro insieme.
02 - 49	sono assegnati ai campi e sottocampi che compongono la struttura o il record. Si scrivono a partire dal margine B
77	è utilizzato per identificare dati indipendenti (indici, contatori). Si scrive a partire dal margine A
88	è utilizzato quando si vuole assegnare un nome a uno o più valori che un dato può assumere Si scrive a partire dal margine B

DESCRIZIONE DATO

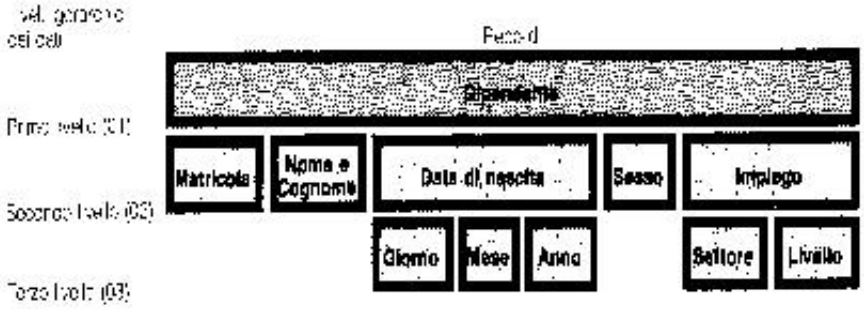
Numero di livello	Nome Variabile	PICTURE Quantità caratteri o cifre

01 PIPPO PIC X(8)

DATA DIVISION - 2

La **WORKING-STORAGE SECTION** è la sezione della DATA DIVISION in cui vengono descritti tutti i dati e le aree di lavoro utilizzate, nel corso del programma,

DEFINIZIONE E STRUTTURA DI UN RECORD



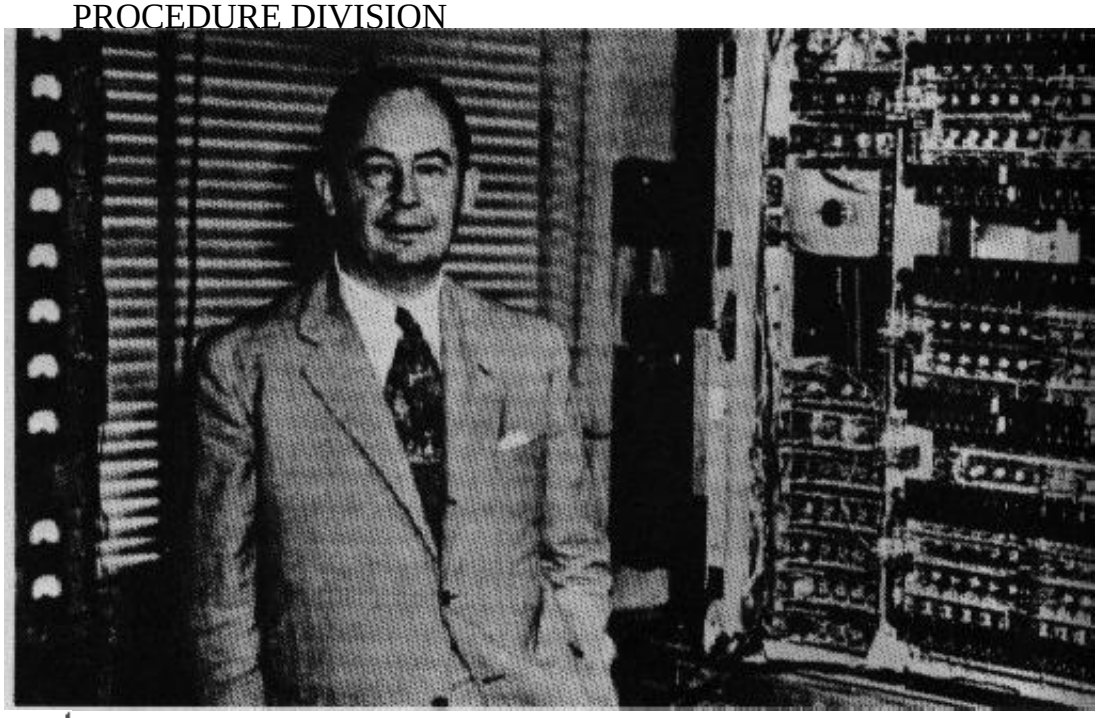
Codifica COBOL della struttura

```
    A    B
IDENTIFICATION DIVISION.
PROGRAM-ID. nome-programma.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. nome-computer.
OBJECT-COMPUTER. nome-computer.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  DIPENDENTE.
    02 MATRICOLA          PIC 999.
    02 NOME-COGNOME       PIC X(30).
    02 DATA-NASCITA.
      03 GIORNO           PIC 9(2).
      03 MESE             PIC 9(2).
      03 ANNO             PIC 9(2).
    02 SESSO              PIC X.
    02 IMPIEGO.
      03 SETTORE          PIC 9(3).
      03 LIVELLO          PIC 9(3).
```

PROCEDURE DIVISION



La sua struttura, a differenza delle altre DIVISION, è molto elastica ed è il programmatore stesso che la stabilisce, in relazione alle esigenze di programmazione

Può essere divisa in SECTION ognuna delle quali deve essere contraddistinta da un nome, scelto dal programmatore, seguito dalla parola SECTION e da un punto finale (il tutto scritto a partire dal margine A)

Ogni SECTION deve contenere almeno un Paragrafo. Ogni paragrafo è identificato da un nome, meglio se significativo, scelto dal programmatore seguito da un punto e scritto a cominciare dal margine A

Ogni paragrafo può contenere una o più istruzioni scritte a partire dal margine B.

(nella foto lo scienziato ungherese J.von Neumann)

N.Brugaletta

COBOL 8

PRIME ISTRUZIONI COBOL

Le istruzioni (**statements**) cominciano sempre con una parola chiave chiamata VERBO e finiscono con un punto, lo spazio, un terminatore.

La sequenza di una o più istruzioni terminante con un punto prende il nome di *periodo* (**sentence**)

Lettura-scrittura da terminale

ACCEPT Permette di leggere ciò che viene digitato sulla tastiera e di memorizzarlo in una variabile.

Es:

```
ACCEPT NOME-CLIENTE
```

DISPLAY Serve a trasferire su video il contenuto di una o più variabili e/o di una o più costanti

Es:

```
DISPLAY "totale " TOT
```

Terminazione di un programma

STOP RUN È l'istruzione che chiude tutti i programmi COBOL esclusione fatta per quelli richiamati da altri programmi cui devono ripassare il controllo

ISTRUZIONI ARITMETICHE - 1

ADD Permette di sommare due o più operandi e di conservare il risultato in una variabile.

FORMATI

ADD identif1 identif2 ... TO identifn
ADD identif1 identif2 ... GIVING identifn
ADD CORR nome-gruppo1 TO nome-gruppo2

SUBTRACT Permette di sottrarre il valore di un campo da uno o più campi e di conservare il risultato.

FORMATI

SUBTRACT identif1 identif2 ... FROM identifn
SUBTRACT identif1... FROM identif2 GIVING identifn
SUBTRACT CORR nome-gruppo1 FROM nome-gruppo2

MULTIPLY Permette di moltiplicare i valori di due campi e di conservare il risultato.

FORMATI

MULTIPLY identif1 BY identif2 ...
MULTIPLY identif1 BY identif2 GIVING identif3

ISTRUZIONI ARITMETICHE - 2

DIVIDE Per dividere due numeri fra di loro è possibile specificare il dividendo e il divisore in vari modi

FORMATI

DIVIDE divisore INTO dividendo
DIVIDE dividendo BY divisore GIVING quoziente ...
... REMAINDER resto

COMPUTE Il verbo permette di costruire una istruzione per il calcolo di espressioni aritmetiche

FORMATO

COMPUTE identif = espressione

TRASFERIMENTO DI VALORI

MOVE Il verbo permette di trasferire dati da un campo ad un altro

FORMATI

MOVE identif1 TO identif2 ...
MOVE CORR nome-gruppo1 TO nome-gruppo2

Controllo del Flusso di un programma - 1

LA SELEZIONE

Serve a determinare la strada da far seguire all'algoritmo in dipendenza del verificarsi o meno di una determinata condizione.

Nello standard ANSI-85 è stato aggiunto, rispetto al COBOL tradizionale (ANSI-74), un terminatore

FORMATO

IF condizione
  istruzione/i
ELSE
  istruzione/i
END-IF

La clausola ELSE può mancare e la struttura, se comprende un punto, si intende chiusa e, quindi, non necessita del terminatore END-IF

CONDIZIONE →

EQUAL TO
  =
NOT EQUAL TO
  NOT =
GREATER
  >
NOT GREATER
  NOT >
LESS
  <
NOT LESS
  NOT <

Controllo del Flusso di un programma - 2

SOTTOPROGRAMMI E CICLI

L'istruzione PERFORM permette, nell'ambito del programma, il salto ad una particolare procedura. A procedura ultimata il controllo del programma ritorna all'istruzione immediatamente successiva alla PERFORM.

La procedura può essere eseguita più volte in relazione alla sintassi utilizzata.

La procedura richiamata termina con l'istruzione **EXIT**.

Lo standard ANSI-85 permette un uso *on-line* della PERFORM

FORMATI

PERFORM nome-section
PERFORM nome-section UNTIL condizione-uscita
PERFORM nome-section VARYING contatore ...
FROM inizio BY incremento UNTIL condizione-uscita

Uso *on-line* della PERFORM

```
PERFORM UNTIL condizione-uscita
  istruzione
  istruzione
END-PERFORM
```

Esempio completo di schema di programma scritto con tecnica modulare

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CALCTOT.
  *  dati l'imponibile e la percentuale IVA
  *  il programma calcola il totale e l'IVA
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. COBOL85.
OBJECT-COMPUTER. COBOL85.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  DATI-FATTURA.
    02 IMPONIBILE   PIC 9(6).
    02 PERC-IVA     PIC 99.
01  DATI-FINALI.
    02 TOTALE       PIC 9(6).
    02 IVA          PIC 9(6).
PROCEDURE DIVISION.
MAIN SECTION.
IN-MAIN.
    PERFORM INIZIO.
    PERFORM ELABORA.
    PERFORM FINE.
FI-MAIN.  STOP RUN.
INIZIO SECTION.
IN-INIZIO.
    DISPLAY "Immettere Imponibile".
    ACCEPT IMPONIBILE.
    DISPLAY "Immettere Percentuale IVA".
    ACCEPT PERC-IVA.
FI-INIZIO.  EXIT.
ELABORA SECTION.
IN-ELABORA.
    COMPUTE IVA = IMPONIBILE * PERC-IVA / 100.
    ADD IMPONIBILE IVA GIVING TOTALE.
FI-ELABORA.  EXIT.
FINE SECTION.
IN-FINE.
    DISPLAY "Totale IVA " IVA.
    DISPLAY "Totale Fattura " TOTALE.
FI-FINE.  EXIT.
```

Controllo del Flusso di un programma - 3

SCELTA MULTIPLA

Il verbo EVALUATE permette di eseguire istruzioni differenti a seconda del risultato che si ottiene in seguito ad una valutazione (sequenza di IF a cascata)

FORMATI

```
EVALUATE identif
    WHEN  valore1  THRU  valore2
        istruzioni
    WHEN  valore3  THRU  valore4
        istruzioni
   ...
   WHEN OTHER
        istruzioni
END-EVALUATE.
```

Esegue le istruzioni specificate di conseguenza ai **valori** assunti da una o più variabili

```
EVALUATE TRUE
   WHEN  condizione1  ALSO  condizione2
        istruzioni
   WHEN  condizione3  ALSO  condizione4
        istruzioni
   ...
   WHEN OTHER
        istruzioni
END-EVALUATE.
```

Esegue le istruzioni specificate di conseguenza alle **condizioni** enunciate

Livello 88 e nomi-condizioni

- Il livello 88 (in Area B) serve per definire i **nomi-condizione**
- Una variabile assume nome diverso in ragione del valore in essa contenuto
- Viene utilizzato per migliorare la leggibilità del programma

Esempio di utilizzo del livello 88 e di ***test sul nome-condizione***

```
...
01  IMPIEGATO-SCUOLA PIC 9.
    88 BIDELLO       VALUE 1.
    88 APPLICATO     VALUE 2.
    88 PROFESSORE    VALUE 3.
    88 PRESIDE       VALUE 4 THRU 6.
...
PROCEDURE DIVISION.
...
   *    Valuta se c'è il valore 1
    IF BIDELLO
...
   *    Valuta se c'è un valore compreso tra 4 e 6
    IF PRESIDE
...
```

Gestione di Tabelle in COBOL - 1

- La tabella è un **insieme finito** di elementi contenenti **dati omogenei** (stessa descrizione, stessa lunghezza)
- I dati in TABELLA occupano, in memoria, una serie di campi adiacenti
- Si individua un elemento all'interno della Tabella specificando il *subscritto* (**indice**) ad esso associato
- L'indice è un **numero intero positivo diverso da zero** che può essere una costante intera o una variabile di tipo intera

ENIAC, il primo calcolatore elettronico della storia (anno 1946, peso 13 tonnellate)

N.Brugaletta

Gestione di Tabelle in COBOL - 2
metodo subscripting

Esempio di Gestione di un Listino Prezzi

Nº d'ordine	Codice Articolo	Prezzo Unitario	Descrizione
1	61305T	12000	Pialla regolabile
2	61320R	14500	Saldatore elettrico
3	61340M	8900	Cacciavite automatico
..	.....	.....	.....
30	62338T	12000	Pistola a spruzzo

Implementazione della Tabella

```
...
01  LISTINO.
    02 ARTICOLO
OCCURS 30 TIMES.
       03 CODART PIC
X(6).
       03 PZUNIT PIC
9(6).
       03 DESCRI PIC
X(30).
    02 QUANT    PIC
99 VALUE ZERO.
...
```

La **riga Articolo** verrà ripetuta **al massimo** 30 volte

Quantità **effettiva** delle righe presenti nella Tabella (al momento attuale non esistono

Gestione di Tabelle in COBOL - 3

Caricamento della Tabella

```
...
CARICA SECTION.
IN-CARICA.
  *    Riceve la quantità degli elementi
  *    da caricare in tabella (Massimo 30)
     ACCEPT QUANT.

  *    Carica la tabella con i dati di input
     PERFORM VARYING CONTA
           FROM 1 BY 1 UNTIL CONTA > QUANT
       ACCEPT CODART (CONTA)
       ACCEPT PZUNIT (CONTA)
       ACCEPT DESCRI (CONTA)
     END-PERFORM.
FI-CARICA.  EXIT.
...
```

Gli Elementi della Tabella

Si può fare riferimento alla **riga della tabella** nel suo complesso, come anche ai **singoli attributi** dell'elemento. In ogni caso **deve sempre** essere specificato l'indice:

```
DISPLAY DESCRI (2)
```

visualizza

Saldatore elettrico

```
DISPLAY ARTICOLO (2)
```

visualizza

61320R014500Saldatore elettrico

Frammento di programma di gestione della Tabella con uso del livello 88

```
      *    Programma per la ricerca di un elemento
      *    all'interno di una tabella
...
WORKING-STORAGE SECTION.
...
      *     variabile logica per la ricerca
01  CERCA-IN-TAB   PIC 9.
    88 TROVATO     VALUE 1.
    88 NON-TROVATO VALUE 0.
      *     codice articolo da cercare in tabella
01  COD-DA-INP     PIC X(6).
...
PROCEDURE DIVISION.
MAIN SECTION.
IN-MAIN.
    PERFORM DATI-INIZIALI.
    MOVE 1 TO CONTA.
    SET NON-TROVATO TO TRUE.
    PERFORM CERCA UNTIL TROVATO OR CONTA > QUANT.
    PERFORM RISULTATO.
FI-MAIN.  STOP RUN.
DATI-INIZIALI SECTION.
IN-DATINIZ.
    PERFORM CARICA.
    ACCEPT COD-DA-INP.
FI-DATINIZ.  EXIT.
CARICA SECTION.
IN-CARICA.
...
FI-CARICA.  EXIT.
CERCA SECTION.
IN-CERCA.
   *    Verifica se il codice proveniente da input è uguale
   *    al codice selezionato della tabella
    IF COD-DA-INP = CODART (CONTA)
      SET TROVATO TO TRUE
    ELSE
      ADD 1 TO CONTA
    END-IF.
FI-CERCA.  EXIT.
RISULTATO SECTION.
IN-RISUL.
    IF TROVATO
      DISPLAY "Elemento trovato in posizione " CONTA
    ELSE
      DISPLAY "Elemento non presente in tabella"
    END-IF.
FI-RISUL.  EXIT.
```

Il COBOL e la Gestione dei Files - 1

ORGANIZZAZIONE dei Files
(Sistemazione dei records all'interno del File)

Viene effettuata in funzione del modo con cui i records dovranno essere elaborati

SEQUENTIAL

I records sono registrati uno dopo l'altro, nello stesso ordine con cui si presentano al terminale di entrata

INDEXED

I records sono registrati in modo che ognuno sia individuato da una chiave principale e, nel caso, da chiavi secondarie. A ogni chiave è associato un indice

RELATIVE

I records sono individuati dal *numero* stesso del record (la sua posizione fisica all'interno del File)

Modalità di ACCESSO
(Come i records verranno letti o scritti)

Varia in funzione del tipo di elaborazione da svolgere sui records

SEQUENTIAL
(Coda-Elaborazione di **tutti** i records)

I records sono processati nell'*ordine con cui sono stati* originariamente *scritti* (organizzazione Sequential), in *ordine ascendente* di *chiave* (org. Indexed), in *ordine ascendente* di *numero* di record (org. Relative)

RANDOM
(Tabella-**Singoli** records indipendentemente dall'ordine)

È consentito su files Indexed e Relative. I records da processare sono rintracciati specificando la chiave o il numero del record

DYNAMIC
(Selezione di records con determinate caratteristiche)

Non gestibile su files Sequential, consente di variare il modo di accesso (posizionamento random, elaborazione sequential)

Il COBOL e la Gestione dei Files - 2

le COMBINAZIONI AMMESSE

Organ. / Accesso	Sequential	Indexed	Relative
Sequenziale	si	si	si
Casuale		si	si
Dinamico		si	si

Gestione dei Files

Per poter gestire i files in un programma COBOL è necessario seguire **rigidamente** i passi seguenti

1) SELEZIONE DEL FILE
2) DESCRIZIONE DEL FILE E DEL TRACCIATO RECORD
3) APERTURA DEL FILE
4) ELABORAZIONE DEI RECORDS
5) CHIUSURA DEL FILE

Il non seguire i suddetti passi può comportare disfunsioni.

Files ad organizzazione SEQUENTIAL - 1

SELEZIONE DEL FILE

Si debba elaborare l'archivio assicurati di una compagnia di assicurazioni:

```
...
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
...
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT ASSICURATI ASSIGN TO DISK "ASSICURA.DAT"
        ORGANIZATION SEQUENTIAL
        ACCESS MODE  SEQUENTIAL.
...
```

Il *periodo* SELECT associa il nome **logico** ASSICURATI al nome **fisico** ASSICURA.DAT (il nome con cui è registrato il file su memoria di massa): all'interno del programma il file sarà identificato da tale nome.

Le clausole ORGANIZATION e ACCESS MODE possono, in questo caso, essere omesse, in quanto l'organizzazione e la modalità di accesso SEQUENTIAL sono assunte per **default**

DESCRIZIONE FILE E TRACCIATO RECORD

```
...
DATA DIVISION.
FILE SECTION.
FD  ASSICURATI LABEL RECORD STANDARD.
01  R-ASSICURATI.
    02 POLIZZA   PIC X(10).
    02 NOME      PIC X(40).
    02 ETA       PIC 999.
    02 SESSO     PIC X.
    02 RESIDENZA PIC X(15).
    02 SCADENZA  PIC X(6).
WORKING-STORAGE SECTION.
...
```

Nella FILE SECTION ci saranno tante FD (File Description) quanti sono i files definiti nella FILE-CONTROL.

La lettura di un record dal file permetterà di depositare nell'area R-ASSICURATI i dati conservati, così come la scrittura trasferirà su disco il contenuto della stessa area

Files ad organizzazione SEQUENTIAL - 2

APERTURA DEL FILE

Per poter utilizzare un file è necessario stabilire un canale di comunicazione fra il file su memoria di massa e l'area di memoria centrale preparata per ricevere i dati

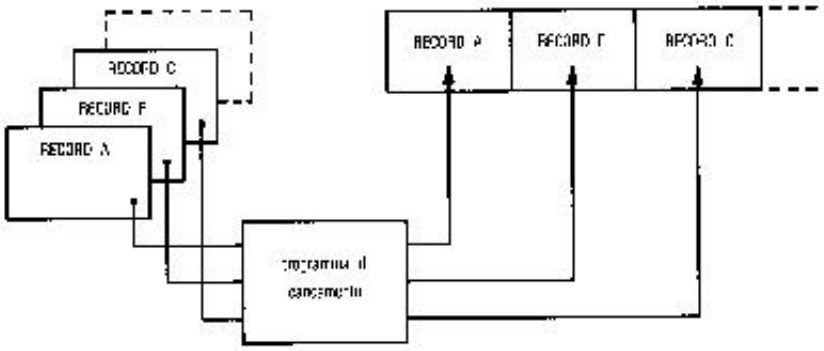
...
PROCEDURE DIVISION.
...
OPEN modalità di apertura nome file

La OPEN abilita la comunicazione con la memoria di massa.
Il nome del file di cui si parla è quello **logico** stabilito nella SELECT

Modalità di Apertura →

INPUT	Il file viene predisposto per la **lettura**: il puntatore di file viene posizionato all'inizio
OUTPUT	Il file viene predisposto per la **scrittura**: il puntatore di file viene posizionato all'**inizio**
EXTEND	Il file viene predisposto per la **scrittura**: il puntatore di file viene posizionato **in coda** all'ultimo record inserito

Files ad organizzazione SEQUENTIAL - 3



ELABORAZIONE DEI RECORDS

READ

Copia il record, attualmente puntato, dalla memoria di massa alla apposita area. Il puntatore viene spostato sul prossimo record

FORMATO

READ nome file NEXT RECORD
  AT END
    istruzione/i
  NOT AT END
    istruzione/i
END-READ

La clausola AT END specifica cosa fare se si è arrivati alla fine del file, viceversa la NOT AT END (opzionale)

WRITE

Scrive su memoria di massa il record attualmente nell'area di comunicazione (il file deve essere aperto in OUTPUT o EXTEND)

FORMATO

WRITE nome record

L'istruzione di scrittura fa riferimento al nome del record, quella di lettura al nome del file

Files ad organizzazione SEQUENTIAL - 4

CHIUSURA DEL FILE

Finito di lavorare sul file è necessario chiuderlo, dissociarlo dal canale di comunicazione con la memoria di massa.

CLOSE nome file

Tale istruzione **è unica per tutti i tipi di organizzazione dei files**: l'apertura deve indicare il senso del traffico dei dati tra memoria centrale e memorie di massa, la chiusura indica più semplicemente l'interruzione del canale di comunicazione

N.Brugaletta

Esempio di programma di gestione di un File SEQUENTIAL

```
      *    Ricerca delle polizze in scadenza ad una determinata data
...
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ASSICURATI ASSIGN TO DISK "ASSICURA.DAT".
DATA DIVISION.
FILE SECTION.
FD  ASSICURATI LABEL RECORD STANDARD.
01  R-ASSICURATI.
    02 POLIZZA   PIC X(10).
    02 NOME      PIC X(40).
    02 ETA       PIC 999.
    02 SESSO     PIC X.
    02 RESIDENZA PIC X(15).
    02 SCADENZA  PIC X(6).
WORKING-STORAGE SECTION.
01  DATA-SCADUTE PIC X(6).
   *    Controllo se ci sono ancora record da elaborare
01  CONTR-FILE     PIC 9.
    88 FINE-FILE   VALUE 1.
    88 NOFINE-FILE VALUE 0.
PROCEDURE DIVISION.
MAIN SECTION.
IN-MAIN.
    PERFORM INIZIO.
    PERFORM ELABORA UNTIL FINE-FILE.
    PERFORM FINE.
FI-MAIN.  STOP RUN.
INIZIO SECTION.
IN-INIZIO.
    OPEN INPUT ASSICURATI.
    ACCEPT DATA-SCADUTE.
    SET NOFINE-FILE TO TRUE.
    PERFORM LEGGI-POLIZZA.
FI-INIZIO.  EXIT.
ELABORA SECTION.
IN-ELABORA.
    IF SCADENZA = DATA-SCADUTE
      DISPLAY R-ASSICURATI
    END-IF.
    PERFORM LEGGI-POLIZZA.
FI-ELABORA.  EXIT.
FINE SECTION.
IN-FINE.
    CLOSE ASSICURATI.
FI-FINE.  EXIT.
LEGGI-POLIZZA.
IN-LEGGI.
    READ ASSICURATI NEXT RECORD
      AT END
        SET FINE-FILE TO TRUE
    END-READ.
FI-LEGGI.  EXIT.
```

Files ad organizzazione INDEXED - 1

SELEZIONE DEL FILE

Si debba elaborare l'archivio anagrafico dei dipendenti di una ditta:

```
...
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
...
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DIPENDENTI ASSIGN TO DISK "DIPEN.DAT"
        ORGANIZATION INDEXED
        ACCESS MODE  RANDOM
        RECORD KEY   CODICE.
...
```

In questo caso la clausola ORGANIZATION va specificata obbligatoriamente.

La modalità di accesso RANDOM specifica che, nell'elaborazione seguente, i records si susseguiranno con un ordine dipendente soltanto dalla elaborazione stessa.

La clausola RECORD KEY specifica il *campo del record* utilizzato come chiave di accesso al file.

DESCRIZIONE FILE E TRACCIATO RECORD

```
...
DATA DIVISION.
FILE SECTION.
FD  DIPENDENTI LABEL RECORD STANDARD.
01  R-DIPENDENTI.
    02 CODICE    PIC X(2).
    02 NOME      PIC X(15).
    02 COGNOME   PIC X(15).
    02 QUALIFICA PIC X(20).
WORKING-STORAGE SECTION.
...
```

Nel tracciato record *deve* esistere, così come specificato nella FILE-CONTROL, il campo CODICE.

Il campo utilizzato come chiave *può* essere sottostrutturato e *deve* essere di tipo alfanumerico

Files ad organizzazione INDEXED - 2

APERTURA DEL FILE

L'apertura del file INDEXED segue le stesse modalità del file SEQUENTIAL. Cambiano, almeno in parte, le modalità di apertura:

```
...
PROCEDURE DIVISION.
...
```

OPEN modalità di apertura nome file

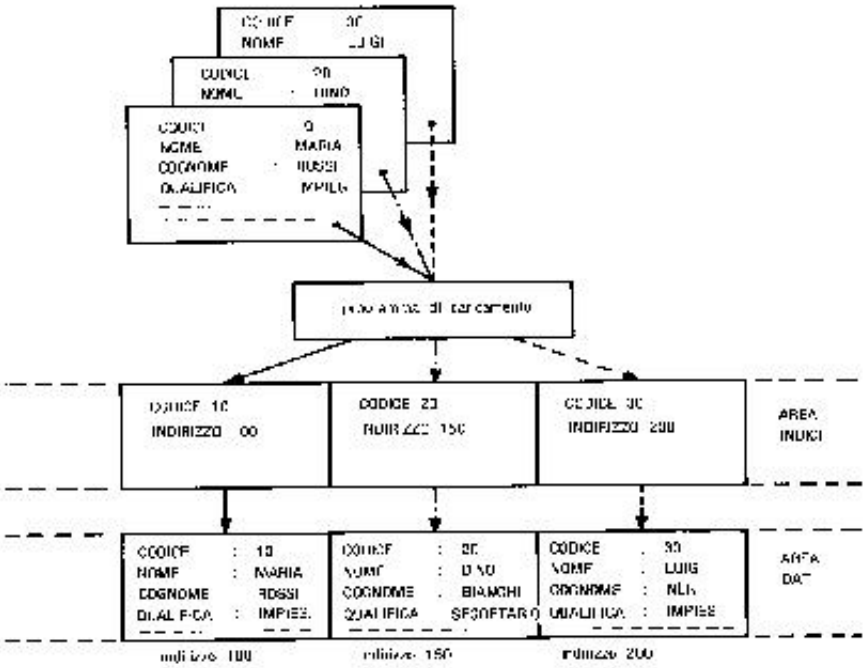
La OPEN abilita la comunicazione con la memoria di massa.

Il nome del file di cui si parla è quello **logico** stabilito nella SELECT

Modalità di Apertura →

INPUT	Il file viene predisposto per la **lettura**: i records del file possono essere solo letti
OUTPUT	Il file viene predisposto per la **scrittura**: i records del file possono essere solo scritti
I-O	Il file viene predisposto in **aggiornamento**: i records possono essere scritti, letti e modificati

Files ad organizzazione INDEXED - 3



ELABORAZIONE DEI RECORDS - 1

I records del file sono associati alla *chiave*: qualunque operazione debba essere effettuata su di essi presuppone che tale *chiave* sia specificata.

READ

Copia il record, associato alla chiave specificata, dalla memoria di massa alla apposita area.

FORMATO

READ nome file
  INVALID KEY
    istruzione/i
  NOT INVALID KEY
    istruzione/i
END-READ

La clausola INVALID KEY specifica cosa fare se ***non esiste*** il record associato alla chiave specificata, viceversa la NOT INVALID KEY (opzionale).

Si può utilizzare con modalità di apertura INPUT e I-O

ELABORAZIONE DEI RECORDS - 2

WRITE Scrive su memoria di massa il record attualmente nell'area di comunicazione (il file deve essere aperto in OUTPUT o I-O)

FORMATO

WRITE nome record
  INVALID KEY
    istruzione/i
  NOT INVALID KEY
    istruzione/i
END-WRITE

La clausola INVALID KEY specifica cosa fare se ***esiste già*** un record associato alla chiave specificata, viceversa la NOT INVALID KEY (opzionale).

Si può utilizzare con modalità di apertura INPUT e I-O

ELABORAZIONE DEI RECORDS - 3

REWRITE Aggiorna il record associato alla chiave *attualmente* specificata (il file deve essere aperto in I-O)

FORMATO

REWRITE nome record
  INVALID KEY
    istruzione/i
  NOT INVALID KEY
    istruzione/i
END-REWRITE

La clausola INVALID KEY specifica cosa fare se ***non esiste*** un record associato alla chiave specificata, viceversa la NOT INVALID KEY (opzionale).

Si può utilizzare con modalità di apertura INPUT e I-O

Esempio di programma di gestione di un File INDEXED

```
      *    Visualizzazione Dati Dipendenti associati a determinati Codici
...
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DIPENDENTI ASSIGN TO DISK "DIPEN.DAT"
         ORGANIZATION INDEXED
         ACCESS MODE  RANDOM
         RECORD KEY   CODICE.
DATA DIVISION.
FILE SECTION.
FD  DIPENDENTI LABEL RECORD STANDARD.
01  R-DIPENDENTI.
    02 CODICE    PIC X(2).
    02 NOME      PIC X(15).
     02 COGNOME   PIC X(15).
    02 QUALIFICA PIC X(20).
WORKING-STORAGE SECTION.
01  COD-CERCA    PIC X(2).
    *    Controllo se ci sono ancora dipendenti da elaborare (Codice <> Space)
    *    e se Trovato dipendente con il codice specificato
01  CONTR-FINE     PIC 9.
    88 FINE-ELAB   VALUE 1.
    88 NOFINE-ELAB VALUE 0.
01  CONTR-DIP      PIC 9.
    88 TROVATO     VALUE 1.
    88 NON-TROVATO VALUE 0.
PROCEDURE DIVISION.
MAIN SECTION.
IN-MAIN.
    PERFORM INIZIO.
    PERFORM ELABORA UNTIL FINE-ELAB.
    PERFORM FINE.
FI-MAIN.  STOP RUN.
INIZIO SECTION.
IN-INIZIO.
    OPEN INPUT DIPENDENTI.
    SET NOFINE-ELAB TO TRUE.
    PERFORM RICEVI-COD.
FI-INIZIO.  EXIT.
ELABORA SECTION.
IN-ELABORA.
    SET TROVATO TO TRUE.
    PERFORM CERCA-DIP.
    IF TROVATO
      DISPLAY R-DIPENDENTE
    END-IF.
    PERFORM RICEVI-COD.
FI-ELABORA.  EXIT.
FINE SECTION.
IN-FINE.
    CLOSE DIPENDENTI.
FI-FINE.  EXIT.

RICEVI-COD SECTION.
IN-RCOD.
    ACCEPT COD-CERCA.
    IF COD-CERCA = SPACE
      SET FINE-ELAB TO TRUE
    END-IF.
FI-RCOD.  EXIT.

CERCA-DIP SECTION.
IN-CERDIP.
*   Inizializza Chiave
    MOVE COD-CERCA TO CODICE.
*   Cerca Dipendente con
*   quella chiave
    READ DIPENDENTI
      INVALID KEY
        SET NON-TROVATO TO TRUE
    END-READ.

FI-CERDIP.  EXIT.
```