Introduzione al C

Linguaggi di programmazione

Sono linguaggi che permettono la codifica di algoritmi in modo da renderli eseguibili, direttamente o indirettamente, dall'elaboratore.
Si dividono in:

- Linguaggi macchina.
- Linguaggi assembler.
- Linguaggi di alto livello

Linguaggi macchina

- Le istruzioni sono sequenze di bit e, tipicamente sono composte da un codice operativo e da zero o più operandi.
  - Le istruzioni sono direttamente eseguibili dalla CPU.
  - Ogni CPU ha un proprio linguaggio macchina.

010000	10000	00001	Copia il valore della cella 16 nel registro RA
011000	00001	00001	Incrementa il valore di RA di uno
010001	00001	01111	Copia RA nella cella 15

Linguaggi assembler

- Le istruzioni sono sequenze di caratteri alfa-numerici (le istruzioni in linguaggio macchina sono sostituite da codici mnemonici).
  - Sono eseguibili dall'elaboratore previa la (semplice) traduzione in linguaggio macchina mediante il programma **assemblatore**.
  - Ogni CPU ha un proprio linguaggio assembler.

```
MOV    AX, [16]    Copia il valore della cella 16 nel registro RA
ADD    AX, 1       Incrementa il valore di RA di uno
MOV    [15], AX    Copia RA nella cella 15
```

Linguaggi di alto livello

- Le istruzioni sono sequenze di caratteri alfa-numerici.
  - Rispetto ai linguaggi di basso livello, essi sono "più vicini" al linguaggio naturale. Offrono un'astrazione del calcolatore in modo da rendere il programma indipendente dai dettagli architetturali.
  - Il programma ad alto livello non è direttamente eseguibile dall'elaboratore, ma è necessario utilizzare un **programma di supporto** per poterlo "eseguire".

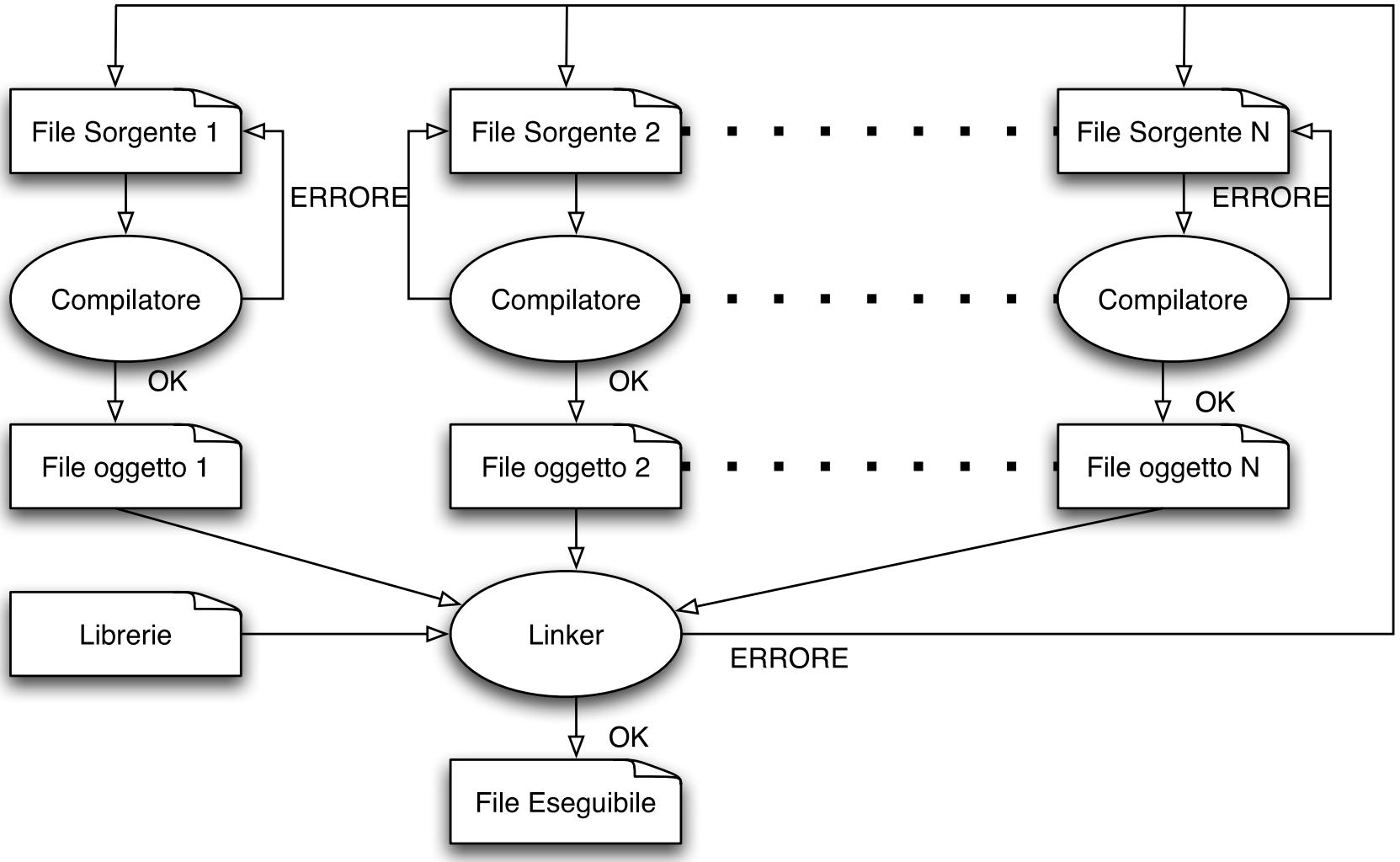
PASCAL	LISP
B:= A + 1	(+ A 1)

Interprete e compilatore

Dato un programma P scritto in un linguaggio di alto livello, esistono due modi per eseguire P:

- Interprete: è un programma che interpreta ed esegue ogni istruzione I di P secondo la semantica operazionale di I.
- Compilatore: è un programma che traduce P in un **equivalente** programma P' scritto in linguaggio macchina. L'esecuzione di P corrisponde, quindi, ad eseguire P'.

Processo di compilazione



Struttura di un programma C

Un programma C si compone delle seguenti parti:

- Direttive per il preprocessore.
- Definizioni di:
  - nuovi tipi,
  - costanti,
  - variabili e
  - funzioni.
- Dichiarazioni di:
  - variabili e
  - funzioni.

Esempio

```
#include <stdio.h> /* dir. preprocessore */

typedef int T[10]; /* def. di tipo */

const double pi=3.1415; /* def. di costante */

int y; /* def. di variabile */

int quad(int); /* dich. di funzione */

int main (void) /* def. di funzione */
{
    int x;
    x = 5;
    printf("Il quadrato di x: %d", quad(x));
    return 0;
}
```

Nota bene

- Ogni programma C deve definire **una sola** funzione `main`.
- La funzione `main` deve essere definita nel seguente modo:

```
int main (void)
{
      ...

   return 0;
}
```

- Inizialmente i nostri programmi conterranno solo la funzione: `main`.

Struttura della funzione `main`

La funzione `main` si compone di

- L'intestazione.
- Il corpo (compreso tra la parentesi graffa aperta e la corrispondente chiusa) nel quale sono contenute le istruzioni che permettono di calcolare la funzione.

```
int main(void)
{
    int x;
    x = 5;
    printf("Il quadrato di x: %d", quad(x));
    return 0;
}
```

Definizione di variabile locale

La definizione di variabile locale (`int x;`) è una istruzione che permette di riservare durante l'esecuzione della funzione lo spazio in memoria centrale per la variabile.

- Le definizioni devono essere scritte all'inizio del corpo.
- La definizione si compone di due parti:
  1. **tipo** della variabile (`int`);
  2. **nome** della variabile (`x`).

La dimensione dello spazio di memoria centrale riservato per la variabile dipende dal suo tipo.

Tipo di dato

- Il tipo di dato determina l'insieme di valori e le operazioni che si possono compiere sui valori.
- Ad esempio: il tipo intero è composto dai numeri interi e dalle usuali operazioni aritmetiche.
- Il tipo `int` del C è un'approssimazione degli interi mediante la rappresentazione in complemento a due ed il numero di bit utilizzati dipende dal compilatore.
- Il tipo `double` del C è un'approssimazione dei reali mediante la rappresentazione in virgola mobile (formato IEEE 754) ed il numero di bit utilizzati per la matissa ed esponente dipende dal compilatore.
- **ATTENZIONE:** il C fa differenza tra le lettere maiuscole e quelle minuscole. Ad esempio: `int`, `Int` ed `INT` hanno significato diverso.

Nomi: identificatori

- Un identificatore è una sequenza di lettere o cifre, puÚ contenere il carattere **underscore** (_), ed inizia con una lettera o con lo underscore, ed al massimo puÚ essere composto da 31 caratteri.
  - `x`, `a` e `_acP12` sono identificatori.
  - `1A`, `A?b`, `A\ B` e `A%` non sono identificatori.
- Ogni oggetto in C (variabile, funzione, nuovo tipo, . . .) per essere riferito deve essere nominato. Ogni nome è identificatore.
- L'utente ha la possibilità di scegliere qualsiasi nome per gli oggetti da lui definiti con l'eccezione dei nomi utilizzati dal linguaggio (parole chiave). Ad esempio `int` è una parola chiave.

Come visualizzare dati su video

- La funzione `printf()` (**formatted print**) permette di visualizzare su video una *stringa* contenuta fra doppi apici.
  Ad esempio con printf(”aB12”); viene visualizzato: `aB12`.
- La stringa puÚ contenere delle *specifiche di conversione* che iniziano con il carattere % e ciascuna viene associata a un parametro.
- I parametri sono scritti dopo la stringa e sono separati dalla virgola.
- La specifica di conversione determina il formato secondo cui il corrisponente parametro viene stampato. printf(”Stampa cinque: %d”, 5); viene visualizzato: `Stampa cinque:  5`.
- Per potere utilizzare la funzione `printf` (o una qualsiasi funzione di input/output) è necessasrio inserire nel programma la direttiva del preprocessore

```
#include<stdio.h>
```

Commenti

- La sequenza di caratteri compresa tra `/*` ed `*/` è un commento.
- Un commento si puÚ estendere su più righe e puÚ contenere qualsiasi sequenza di caratteri (anche parole chiave).
- Un commento viene ignorato durante la fase di compilazione.

Esempio

```
#include <stdio.h> /* Questo è un commento */
int main (void)
{
   printf("Il mio primo programma");
   return 0;
}
```