Fondamenti di Informatica

Ingegneria Clinica

Lezione 19/10/2009

Prof. Raffaele Nicolussi

FUB - Fondazione Ugo Bordoni

Via B. Castiglione 59 - 00142 Roma

Docente	**Raffaele Nicolussi**	*rnicolussi@fub.it* *0654803323*
Lezioni Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	**Lunedì, Giovedì, Venerdì**	**12:00 – 13:30**
Esercitazioni Aula 15 Via Tiburtina, 205	**Giovedì**	**14:00 – ???**
Ricevimento:	**Per appuntamento**	**in FUB, per email, per telefono**
Sito web:	**http://w3.uniroma1.it/IngClinFondinf**	

AVVISO

- La lezione di laboratorio di **Giovedì 22 Ottobre** si terrà alle ore **17** in **Via Tiburtina 205**, Aula 15
- La lezione sarà per **entrambi i canali** A-L e M-Z
- Si raccomanda di portare i notebook per chi ne fosse provvisto
- Installare il DevCpp prima del laboratorio

Istruzioni di lettura e scrittura

- **Istruzione di scrittura**

  stampa (<identificatore>); *stampa (somma);*

  printf (<identificatore>); *printf (A);*

- **Istruzione di lettura**

  leggi (<identificatore>); *leggi (i);*

  scanf (<identificatore>); *scanf (D);*

Istruzione di assegnazione

<identificatore> = <espressione>;

somma = somma+i;

1. Viene calcolato il valore della espressione che segue il simbolo = (detto "simbolo di assegnazione")
2. Tale valore viene assegnato alla variabile scritta prima del simbolo di assegnazione
   - E' un errore grave fare l'inverso.

Istruzione di controllo: Istruzione composta


```
{
    <istruzione1>;
    <istruzione2>;
    ……..……
    <istruzioneN>;
    }
```

- Ogni *<istruzione>* può essere una qualsiasi istruzione ammessa nel linguaggio
- Può anche essere un'istruzione composta
- Semantica:
  - *Esegui <istruzione_i> nell'ordine*
  - *Alla fine, passa alla prima istruzione dopo il blocco*

Istruzione di controllo: Istruzione condizionale


```
if <test> <istruzione>;          if (A==B) printf (B)
      else <istruzione>;                 else printf (A);

if <test> <istruzione>;          if (A==B) printf (A);
```

- Le istruzioni condizionali permettono di ramificare l'esecuzione di un programma a seconda del valore di un test, che può dare come valore vero o falso
- Ogni <istruzione> può essere una qualsiasi istruzione del linguaggio
  - I/O, assegnazione, controllo
- Semantica:
  - *Esegui il test*
  - *Se il test torna il valore vero esegui la prima istruzione*
  - *Se il test non torna vero (torna falso) esegui l'istruzione dopo l'else*

Istruzione di controllo: Istruzione di iterazione

```
while <test>                    while (a<= b)
   <istruzione>;                somma=somma+a;
```

- L' istruzione che segue <test> può essere una qualunque istruzione (anche composta o condizionale o di iterazione) ed è detta "corpo" del ciclo.
- Semantica:
  - *Valuta <test>*
  - *Se <test> è vero, esegui <istruzione>*
    - *Altrimenti vai all'istruzione successiva il while*

Implementazione delle istruzioni di iterazione

- Ogni linguaggio ha diverse strutture per implementare l'iterazione:
  - per esempio l'istruzione iterativa in C ha tre varianti:

  - *for*
  - *while*
  - *do ..while*

Esempio

Leggere due numeri e stampare il maggiore.

```
{
   scanf (primo);
   scanf (secondo);
   if (primo > secondo)
                printf (primo);
     else
                printf (secondo);
}
```

- Notare l'indentazione:
  - non ci sono regole, ma è opportuna per una maggiore leggibilità del programma
- Questo programma non è ancora eseguibile, essendo solo un frammento
- Manca la dichiarazione dei dati

Livelli di rappresentazione

- Linguaggi assemblativo e macchina sono troppo difficili per essere usati per scrivere programmi
  - Richiedono conoscenze a basso livello sulla struttura della macchina
- Si usano, perciò, dei Linguaggi ad alto livello adatti per scrivere programmi perché nascondono la complessità all'utente
- Per rendere comprensibili questi programmi alla macchina (che capisce solo il linguaggio macchina) sono necessari dei programmi generali (programmi traduttori, ossia compilatori e interpreti) che effettuano la traduzione da linguaggio ad alto livello a linguaggio macchina
  - Rendono eseguibile il programma scritto in linguaggio ad alto livello
  - Il prodotto finale di un processo di compilazione è un file eseguibile

Livelli di rappresentazione

1. definizione del problema	*Algoritmo*
2. rappresentazione bidimensionale (a vari livelli di dettaglio)	*Liguaggi ad alto livello*
3. utilizzo delle sole istruzioni eseguibili dalla specifica macchina, con codice operativo simbolico e indirizzi simbolici	*Linguaggi assemblativi*
4. istruzioni con codici operativi e indirizzi binari	*Linguaggi assoluti*

- **Un compilatore è un programma che, eseguito su una macchina, legge tutto il programma, scritto in un linguaggio ad alto livello, e produce un corrispondente programma, scritto in linguaggio assoluto, che viene eseguito dalla macchina.**

Ciclo di vita di un programma

1) Descrizione del problema mediante la scrittura di un algoritmo

2) Trasformazione dell'algoritmo in un programma scritto in un linguaggio di programmazione ad alto livello per produrre il codice sorgente
   - Tale codice sarà scritto mediante l'uso di un editor
   - Il programma può essere costituito da uno o più file sorgente.

3) Il file sorgente sarà poi compilato per produrre il file oggetto

4) Se il programma è costituito da più file o se fa uso di file di libreria, il linker si occuperà di collegare i file oggetto per produrre il file eseguibile, ossia il programma vero e proprio.

5) L'ultima fase è quella di caricamento e viene effettuata da un programma chiamato loader: il programma da eseguire viene trasferito in memoria centrale e lanciato.

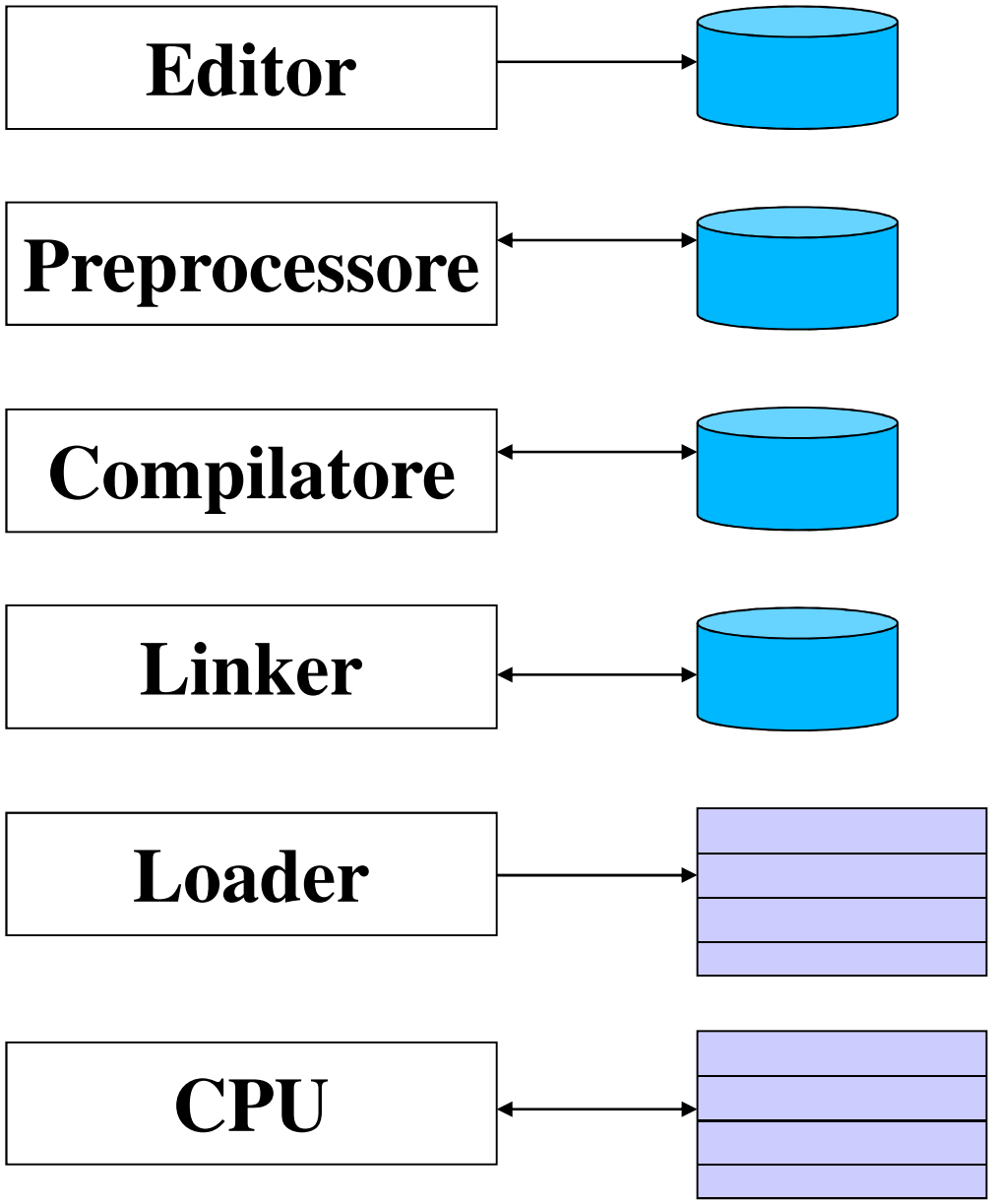
Dal punto di vista dell'operatore (1)

1. **Programma sorgente**: ideato dal programmatore, viene scritto o (se già esistente) modificato usando un editor di testi
2. **Compilatore**: attivato tramite un programma apposito esegue l'analisi del programma sorgente.
   - Possono essere rilevati degli errori
     - Può quindi essere necessaria una modifica del programma per la correzione degli errori e un ritorno al passo 1.
     - In assenza di errori di compilazione, lo traduce in linguaggio macchina

Dal punto di vista dell'operatore (2)

3. **Esecuzione e collaudo: il programma va eseguito e testato per rilevare errori diversi da quelli di compilazione, che possiamo definire di malfunzionamento, che portano ad una correzione del programma e un ritorno al passo 1.**
   - **Errori logici, che portano a risultati non corretti (tipicamente è sbagliato l'algoritmo)**
   - **Errori di run-time (errori al tempo di esecuzione) che portano all'arresto del programma con perdita di quanto inserito o calcolato fino a quel momento (per esempio loop).**
4. **Programma funzionante: al termine del processo, il programma sarà eseguibile e funzionante. Sarà utilizzabile da un utente che fornirà dati di ingresso e ricaverà dati in uscita**

Dal punto di un programma C



Programma creato con un editor e memorizzato nell'hard disk

Il preprocessore esegue il codice

Il compilatore crea il codice oggetto e lo memorizza nell'HD

Il linker collega il codice oggetto con le librerie, crea il .exe e lo memorizza su HD

Il loader carica il programma in memoria centrale

La CPU esegue una istruzione per volta memorizzando i risultati nella memoria centrale

Il linguaggio C (storia-1)

- C linguaggio molto diffuso (anche *C++)*
- Linguaggio portabile: i programmi scritti in C possono essere fatti girare su una qualsiasi macchina che abbia un compilatore C
- La condizione è che si rispetti lo standard C
  - Standard ANSI C
- Sviluppato nel 1972 da Demis Ritchie (AT&T Lab) come linguaggio per scrivere sistemi operativi (OS Unix)
- Per molto tempo l'unico riferimento per il C è stato il libro "The C Reference Manual" di Ritchie
- Nel '77 "The C Programming Language" di Ritchie & Kerningham. che stabilì uno standard di fatto (C K&R)
- Problema: i compilatori aggiungono o tolgono funzionalità al linguaggio base

Il linguaggio C (storia-2)

- Nel 1983 l'ANSI (American Standard Institute) inizia la standardizzazione
- Nel 1989 approvato lo Standard C (ANSI C)
- Altri enti per la standardizzazione lavorarono a questo progetto, tra cui l'ISO (International Standards Organization)
- L'ISO C è uguale all'ANSI C.
- Il linguaggio C++ è una sorta di super insieme dell'ANSI C, a parte alcune caratteristiche del C che non sono incluse
- Se si limita il C ad un semplice sottoinsieme di funzioni (che sono di fatto quelle maggiormente utilizzate) si può scrivere un codice C che può essere compilato da un compilatore C++
- Questo sottoinsieme dell'ANSI è chiamato clean C