



Insegnamento di Programmazione (6 CFU)

Corso di Laurea in Ingegneria Energetica

Introduzione alla programmazione

Ing. Nadia Ranaldo
Dipartimento di Ingegneria
Università degli Studi del Sannio

1

Argomenti

- Cosa significa "programmare"
 - Definizione di algoritmo
 - Rappresentazione di un algoritmo
 - Linguaggi di programmazione
- Compilazione ed interpretazione
- Paradigmi di programmazione
- La storia del C
- Le caratteristiche del C
- La libreria standard del C
- Un tipico ambiente C

2

Cosa significa "programmare"



- Programmazione come attività di **risoluzione di problemi**.
- Comprende varie fasi:
 - Analisi del problema, ad esempio individuare input, output e le loro relazioni;
 - Definizione del procedimento risolutivo (**algoritmo**) per risolvere il problema;
 - Implementazione dell'algoritmo in un linguaggio di programmazione (scrittura del codice);
 - Verifica del programma (testing);
 - Manutenzione ed aggiornamento del programma.
- Nota: La scrittura del codice è solo una piccola fase dell'attività di programmazione!
- **Documentazione** di ciascuna fase

3

Analisi del problema



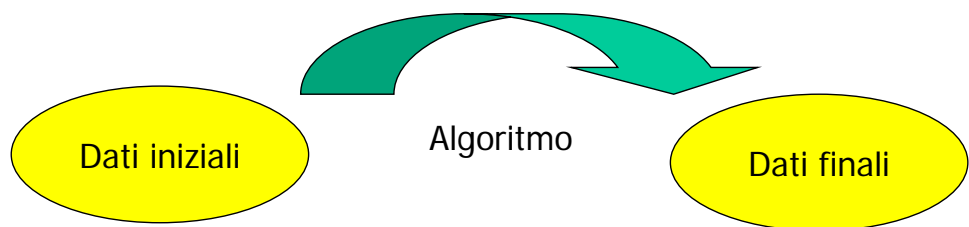
- Capire bene gli obiettivi del problema, ovvero qual è il risultato che si vuole raggiungere
- Quindi occorre evidenziare
 - **le regole da seguire**
 - **i dati espliciti** (indicati in maniera evidente nella traccia del problema) ed **impliciti** (non indicati nella traccia ma necessari a risolvere il problema)
- Eliminare i dettagli inutili ed ambigui

4

Definizione di un procedimento risolutivo

La **descrizione rigorosa** (formale) delle azioni da compiere per risolvere un problema è detta **algoritmo**

- Un algoritmo è la descrizione della soluzione di un problema espressa come un **insieme di regole** che, operando sui dati iniziali, consente di ottenere i risultati del problema
- Tali regole vengono determinate tramite la scomposizione del problema di partenza in problemi sempre più semplici (detti **sottoproblemi**), fino ad arrivare a sottoproblemi elementari, ognuno dei quali è detto passo (**step**) dell'algoritmo

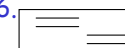


5

Esempi di algoritmi

Procedura per l'ingresso in autostrada

1. **Se** non hai il Telepass vai al passo 2.
Altrimenti vai al passo 6.
 2. **fermati davanti al casello autostradale**
 3. **premi il pulsante**
 4. **prendi e conserva il biglietto, vai al passo 6.**
 5. **rallenta nel pressi della stazione**
 6. **Finchè** la sbarra non si alza vai al passo 7, altrimenti vai al passo 8
 7. **Attendere**, vai al passo 6
 8. **ripartire**
 9. **fine**
- Una ricetta



6

Caratteristiche di un algoritmo

- Lunghezza **finita** dei passi da svolgere, qualunque siano i dati di ingresso (purché si tratti ovviamente di un insieme finito)
- **Struttura in passi elementari**
 - Appartenenti ad un insieme di operazioni fondamentali
 - direttamente eseguibili dall'esecutore
- **Non casualità**
 - risultato certo e ripetibile, ovvero, con gli stessi dati iniziali, si ottiene sempre lo stesso risultato
- **Non ambiguità**
 - Le regole sono interpretate in modo univoco qualsiasi sia l'esecutore dell'algoritmo

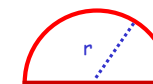
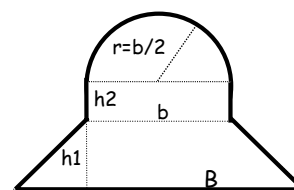
Esempio di algoritmo ambiguo:

bagnare i capelli, applicare lo shampoo, sciacquare e ripetere una seconda volta.

Ripetere cosa??

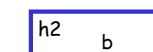
7

Esempio: l'area di una campana



Sottoproblema 1

$$A1 = \frac{1}{2} \pi r^2$$



Sottoproblema 2

$$A2 = b h2$$

Area della campana =

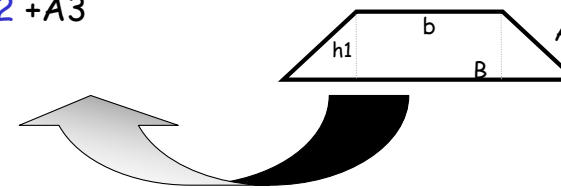
$$A1 + A2 + A3$$

Sottoproblema 3

$$A3 = b h1 +$$

$$\frac{1}{2} \left(\frac{1}{2} (B-b) h1 \right) +$$

$$\frac{1}{2} \left(\frac{1}{2} (B-b) h1 \right)$$



8

Qualità degli algoritmi



Due qualità fondamentali di un algoritmo sono:

- **Correttezza**
 - l'algoritmo permette effettivamente di risolvere il problema in maniera corretta
- **Efficienza**
 - l'esecuzione dell'algoritmo richiede un uso limitato di risorse
 - un algoritmo è tanto più efficiente quanto meno risorse richiede per la sua esecuzione. Una risorsa importante è il **tempo di esecuzione**

9

Sequenza di passi



- Normalmente i passi di un algoritmo sono eseguiti in **sequenza** (ovvero uno dopo l'altro così come sono scritti)
- L'**ordine** dei passi è essenziale per la correttezza dell'algoritmo
- Es. ...
 2. fermati davanti al casello autostradale
 3. premi il pulsante
 4. prendi e conserva il biglietto......
- Non sempre è sufficiente l'esecuzione in sequenza
- E' possibile modificare l'ordine di esecuzione mediante le **strutture di controllo**
 - Sono istruzioni che permettono di esprimere **scelte e ripetizioni**
 - Un algoritmo contenente strutture di controllo può dar luogo a sequenze di esecuzione diverse a seconda dei dati di ingresso

10

Le strutture di controllo: DECISIONE



- Le istruzioni da eseguire sono determinate dalla valutazione di una certa **condizione** che può essere vera o falsa (**condizione booleana**)
 - **SE** è vera una certa condizione
 - **ALLORA**
 - Esegui uno o più passi
 - **ALTRIMENTI**
 - Se la condizione è falsa esegui uno o più passi in alternativa

Es.

- Se non hai il Telepass vai al passo 2. **Altrimenti** vai al passo 6.

11

Le strutture di controllo: ITERAZIONE o RIPETIZIONE



- Le istruzioni vengono eseguite ripetutamente fino a che non si verifica una certa condizione che può essere vera o falsa
 - **MENTRE** è vera una certa condizione
 - Esegui uno o più passi
- Es.
- ...
 - 6. **Finchè** la sbarra non si alza vai al passo 7, altrimenti vai al passo 8
 - 7. Attendere, vai al passo 6
 - ...

12

Rappresentazione degli algoritmi



- Linguaggio naturale
- Diagramma di flusso
- Pseudo-codice
- Linguaggio di programmazione

Rappresentazione degli algoritmi



Linguaggio naturale

- Leggi due numeri
- Calcola la differenza fra i due numeri letti
- Controlla se la differenza e' maggiore di zero
- Se si il massimo e' il primo
- Se no il massimo è il secondo
- Stampa il valore del massimo

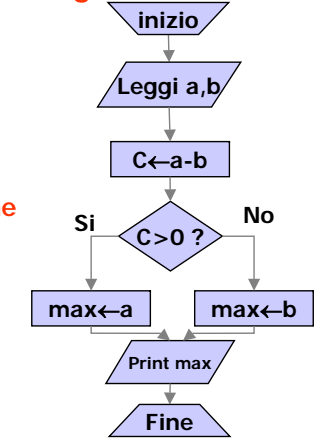
Pseudo-codice

```
input A,B
C ← A - B
if C > 0 then
    max ← A
else
    max ← B
print max
end
```

Linguaggio di programmazione

```
main(String[] args) {
    int A, B,C, max;
    A = Keyboard.readInt();
    B = keyboard.readInt();
    C = A - B;
    if (C > 0)
        max = A;
    else
        max = B;
    System.out.println("Max = "+max);
}
```

Diagramma di flusso

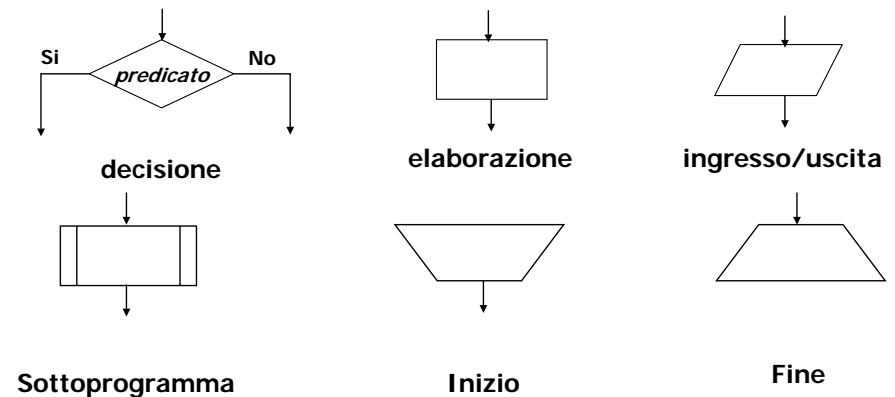


I diagrammi di flusso



- Sono rappresentazioni grafiche dei passi elementari di un algoritmo
- Utili per una visione globale del problema
- Strumento efficace per descrivere un algoritmo (più della descrizione a parole, troppo generica o appesantita da troppi dettagli)
- Le operazioni base sono quattro:
 - Trasferimento di informazioni
 - lettura dati, scrittura risultati, visualizzazione dati intermedi
 - Esecuzione di calcoli (elaborazione)
 - Assunzione di decisioni ed esecuzione di iterazioni
 - Chiamata a sotto-programma (procedura)
- Rappresentate da forme geometriche diverse

Simboli convenzionali

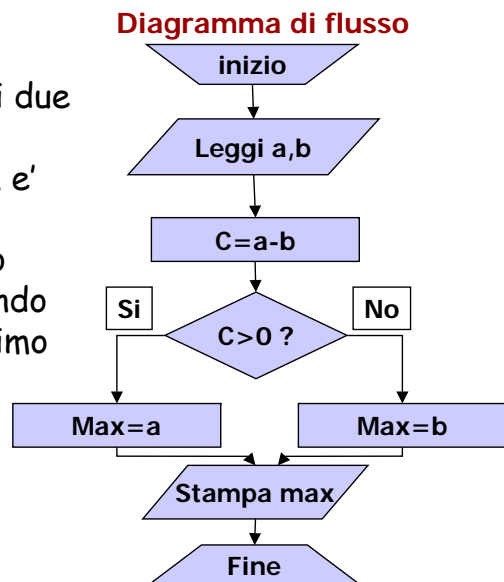


Esempio: massimo di due numeri



Linguaggio naturale

- Leggi due numeri
- Calcola la differenza fra i due numeri letti
- Controlla se la differenza è maggiore di zero
- Se sì il massimo è il primo
- Se no il massimo è il secondo
- Stampa il valore del massimo

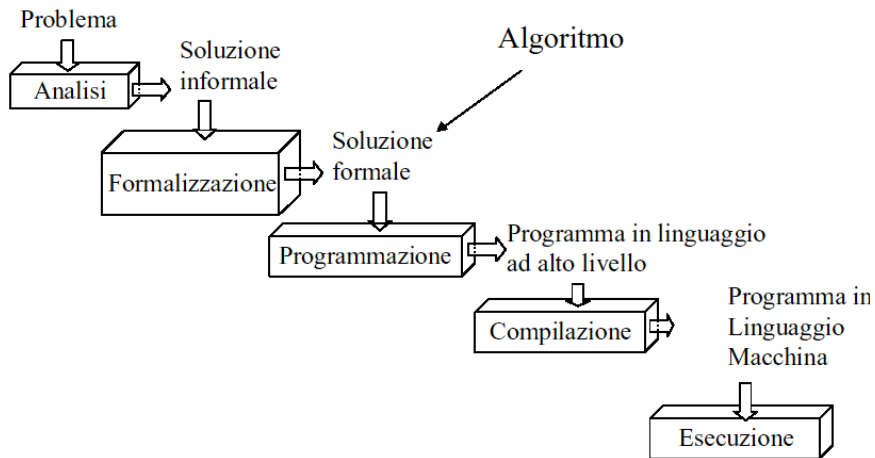


Implementazione dell'algoritmo in un linguaggio di programmazione



- Quando l'esecutore è un computer, gli algoritmi sono detti **programmi**
 - le azioni da eseguire sono dette **istruzioni**
 - il risultato viene anche detto **output del programma**
- Il linguaggio formale utilizzato per scrivere un programma per un calcolatore viene chiamato **linguaggio di programmazione**
- Dopo aver analizzato il problema e aver definito un algoritmo risolutivo si procede con la scrittura del programma in un linguaggio di programmazione

Il processo in dettaglio



Sintassi e semantica



Linguaggio

Sintassi

insieme di regole che consentono di costruire correttamente le frasi del linguaggio

Semantica

disciplina che studia il significato delle parole e delle frasi

LIVELLO	FORMA CORRETTA	FORMA NON CORRETTA
sintattico	sono andato a scuola	ho andato a scuola
semantico	il gatto è un animale	l'albero è un animale

- Definizione di programma: testo (i.e. sequenza di **istruzioni**) scritto in accordo alla **sintassi** e **semantica** di un linguaggio di programmazione



- Un calcolatore è in grado di eseguire direttamente solo programmi scritti nel proprio **linguaggio macchina**, in cui le istruzioni sono scritte in binario
- I programmatori dei primi computer, dovevano descrivere il programma utilizzando direttamente il linguaggio macchina
 - è molto difficile da comprendere per un uomo!
 - è specifico di un computer: computer di tipo diverso hanno linguaggi macchina differenti
- Per rendere più semplice e più rapido il lavoro dei programmatori sono stati introdotti i **linguaggi di programmazione ad alto livello**, ovvero che sono molto più vicini al linguaggio naturale che a quello del computer (fine anni '50)

```
01100010001
00101
11010010010
10101
00101001010
00100
11101001001
10101
.....
```



- Per rendere un programma (scritto in un linguaggio ad alto livello) eseguibile da un computer è necessario tradurre il programma in un programma **equivalente** scritto nel linguaggio macchina del computer
- La traduzione può avvenire in due modi:
 - compilazione
 - interpretazione



- Un programma (detto programma **sorgente**) scritto in un linguaggio di programmazione ad alto livello viene trasformato in un programma in linguaggio macchina (programma **eseguibile**) e **poi può essere eseguito più volte senza dover tradurre nuovamente il programma**



```
0110001000100101
1101001001010101
0010100101000100
1110100100110101
.....
```

controllo della
correttezza sintattica

Nota:
correttezza sintattica



correttezza semantica



- Traduzione riga per riga:
- Ciascuna istruzione del programma scritto in un linguaggio di programmazione ad alto livello viene **trasformata** in istruzioni del linguaggio macchina ed **eseguita**

Compilazione & interpretazione



- Una analogia con la traduzione tra linguaggi diversi
- la compilazione è analoga alla traduzione di un libro
- l'interpretazione è analoga alla traduzione simultanea
- Linguaggi compilati
 - Pascal, C
- Linguaggi interpretati
 - Prolog, Lisp

25

Vantaggi & svantaggi: compilazione



- Vantaggi
 - Creazione di programmi eseguibili in una fase che precede l'esecuzione
 - Elevata velocità d'esecuzione del programma
- Svantaggi
 - Correzioni solitamente al termine della compilazione
 - Impossibilità di controllare il funzionamento solo di una parte del programma
 - Maggior lavoro nella fase di manutenzione
 - ad ogni modifica occorre ricompilare il programma completo

26

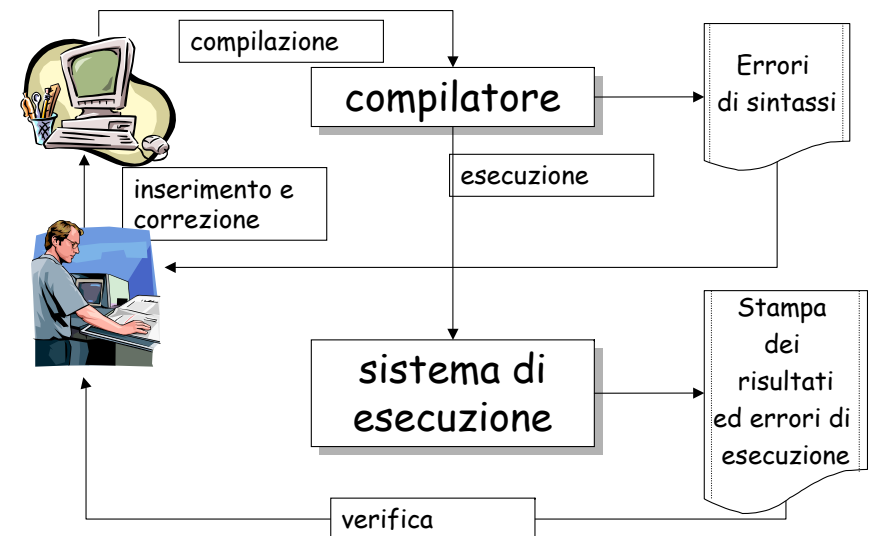
Vantaggi & svantaggi: interpretazione



- Vantaggi
 - Immediata visione dell'errore (l'interprete controlla immediatamente la sintassi)
 - Minor tempo in fase di sviluppo e modifica
 - Possono essere eseguite parti di programmi anche non completi
- Svantaggi
 - Lenta esecuzione
 - Il programma eseguibile dipende dall'interprete perché il codice tradotto non si può memorizzare

27

Verifica del programma (testing)



28

Manutenzione



- Manutenzione del programma =
 - Modificare
 - Aggiornare
 - Estendere
 - Rendere più efficiente
 - ...

29

Paradigmi di Programmazione 1/3



- Definisce/determina il modo in cui il programmatore progetta e scrive il programma
- **Programmazione imperativa**
 - Programma come insieme di istruzioni (direttive o comandi)
 - **Programmazione procedurale (anni '50-'60)**
 - Il punto centrale è l'algoritmo, utilizzato per eseguire una certa elaborazione
 - Una procedura svolge una ben determinata funzione
 - Assembly, Algol, B, BASIC, COBOL, Fortran, etc.
 - **Programmazione strutturata (primi anni '70)**
 - Al crescere delle dimensioni del programma, diventa sempre più difficile gestirlo nel suo complesso
 - Uso di **strutture di controllo** con un ben identificato punto di ingresso e un ben identificato punto di uscita (problemi con l'uso del goto)
 - Almeno una struttura di sequenza, alternativa e iterazione

30

Paradigmi di Programmazione 2/3



- La programmazione strutturata permette di scrivere programmi chiari, corretti e semplici da modificare
- Il primo esempio di linguaggio di programmazione strutturato fu il Pascal, nato nel 1971:
 - Utilizzato per la didattica nella maggior parte delle università
 - Non aveva però molte strutture per renderlo utilizzabile per le applicazioni commerciali, industriali e governative
- Altri esempi: C, Ada, Modula-2, etc.
- **Programmazione dichiarativa o logica (anni '70)**
 - I programmi sono definizioni di **relazioni**
 - Il motore di calcolo è un **dimostratore** di teoremi in grado di effettuare delle deduzioni
 - Utilizzato per sistemi esperti, sistemi di dimostrazione automatica dei teoremi, etc.
 - Prolog, Mercury, etc.

31

Paradigmi di Programmazione 3/3



- **Programmazione funzionale (anni '70)**
 - Il programma è una definizione di funzioni nel senso più **matematico** del termine
 - L'interprete si occupa di **valutare espressioni**, in base alle funzioni di base e a quelle definite dal programmatore
 - Poco diffusi nel campo industriale
 - Lisp, Scheme, etc.
- **Programmazione orientata agli oggetti (OOP) (anni '80)**
 - Concetto chiave di **oggetto**, come metafora del concetto di oggetto del mondo reale
 - Ad ogni oggetto sono associate delle informazioni e delle funzionalità
 - Gli oggetti possono scambiarsi messaggi
 - Un problema è modellato come un insieme di "oggetti software"
 - Java (1994), Smalltalk, C++ (1990), Eiffel, Python, C# (2000), etc.

32

La storia del C



- Sviluppato da Dennis Ritchie da due precedenti linguaggi di programmazione, il BCPL e il B, nel 1972
- E' diventato famoso come linguaggio in cui è stato sviluppato il sistema operativo UNIX
 - Oggi sono scritti in C e C++ quasi tutti i principali sistemi operativi dell'ultima generazione e programmi per il controllo delle periferiche.
- E' indipendente dall'hardware
 - Con una programmazione attenta, è possibile scrivere in C programmi **portabili** sulla maggior parte dei computer.
- Standardizzazione
 - La rapida espansione del C sui diversi tipi di computer ha prodotto molte **varianti** che erano simili, ma spesso incompatibili
 - Divenne allora necessaria una versione standard del C
 - Si creò nel 1983 un comitato tecnico per una definizione del linguaggio che non fosse ambigua e indipendente dalle macchine
 - Lo standard fu approvato nel 1989 e aggiornato nel 1999

33

Le Caratteristiche del C



- Il C è pulito ed efficiente
 - operatori potenti (a livello dei bit)
 - operatori efficienti
 - aritmetica degli indirizzi
- Il C è modulare
 - rende la realizzazione di software di grandi dimensioni più facile
- Il C è alla base del C++
 - ed il C++ è alla base di Java.
- Il C non è un linguaggio perfetto
 - potente ma anche potenzialmente criptico!

34

La libreria standard del C



- I programmi scritti in C, consistono di **moduli** o **pezzi**, chiamati **funzioni**
 - Un programmatore può creare le proprie funzioni
 - I programmatori spesso utilizzano una ricca collezione di funzioni già esistenti, chiamata **libreria standard del C**.
 - è quindi molto importante imparare ad utilizzarle!
 - Utilizzeremo un approccio di costruzione a blocchi per creare i programmi
 - Evitiamo di re-inventare la ruota!
 - se una funzione già esiste, è meglio utilizzarla, piuttosto che scriverne una propria (riusabilità del software)
 - le librerie di funzioni sono scritte con molta cura e sono efficienti, e permettono di scrivere programmi portabili

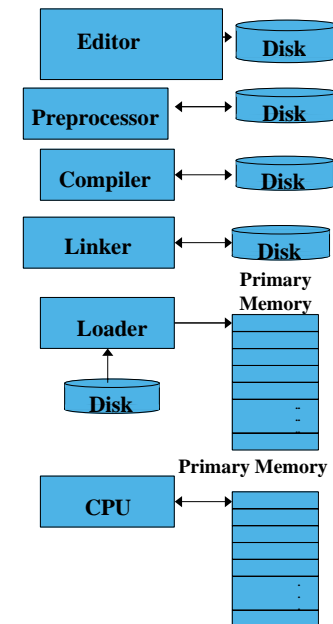
35

Un tipico ambiente C



I programmi scritti in C tipicamente attraversano 6 fasi prima di essere eseguiti in un tipico sistema C

- 1.Edit (editare)
- 2.Preprocess (prelaborare)
- 3.Compile (compilare)
- 4.Link (collegare)
- 5.Load (caricare)
- 6.Execute (eseguire)

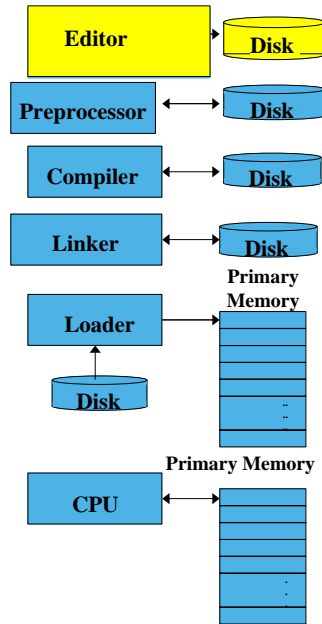


36

Editare

• La prima fase consiste nella scrittura del proprio programma C in un file

- con un programma chiamato **editor**
- se necessario il programmatore eseguirà delle modifiche
- il programma sarà quindi immagazzinato in un dispositivo di memoria secondaria, come l'hard disk.
- il nome del file del programma C dovrà terminare con l'estensione .c
- in Windows il più semplice editor è il Note Pad, ma ce ne sono altri più avanzati!

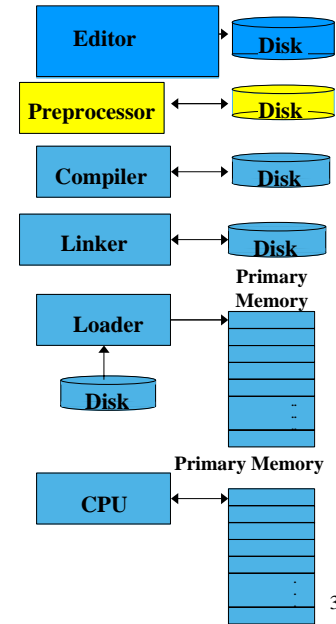


Preelaborare

• In seguito, il programmatore eseguirà il comando di compilazione del programma.

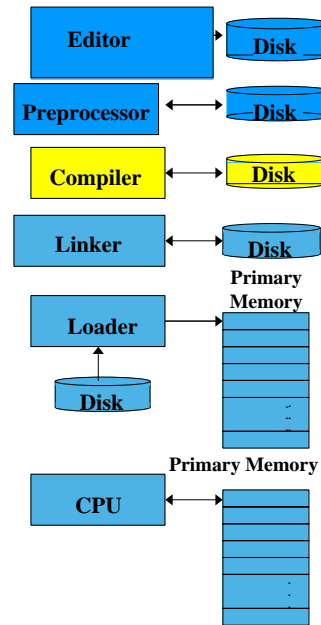
• Prima della fase di traduzione, sarà eseguito automaticamente il **programma preprocessore**

- il preprocessore del C obbedisce a comandi speciali, chiamati **direttive del preprocessore**
- indicano che sul programma dovranno essere eseguite determinate manipolazioni, prima della compilazione vera e propria
- generalmente sono l'inclusione di altri file in quello da compilare e nella sostituzione di simboli speciali con un testo opportuno



Compilare

- Il compilatore traduce il programma C nel codice in linguaggio macchina (detto anche **codice oggetto**)



Fase di linking (collegamento)

I programmi scritti in C contengono tipicamente dei riferimenti a funzioni definite altrove

- per esempio nelle librerie standard

Quindi il codice oggetto prodotto dal compilatore avrà dei "buchi" a causa di queste parti mancanti

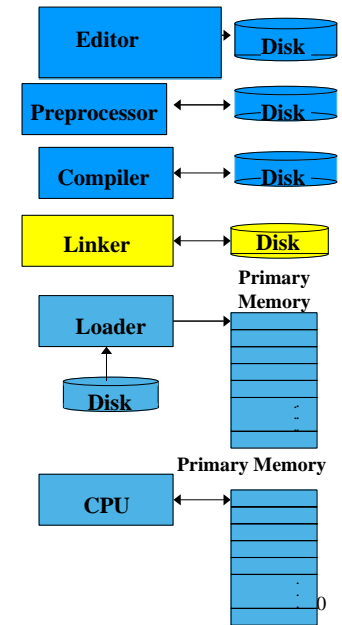
Il linker collega il codice oggetto con quello delle funzioni mancanti per produrre il programma eseguibile

In un sistema tipico basato su Linux, il comando per compilare e collegare i pezzi di un programma è **gcc**

Per esempio per compilare e collegare i pezzi di del programma prova.c bisogna scrivere

gcc prova.c

Se la compilazione e il collegamento non producono errori, sarà prodotto un file chiamato **a.out**, che sarà il programma eseguibile del programma prova.c

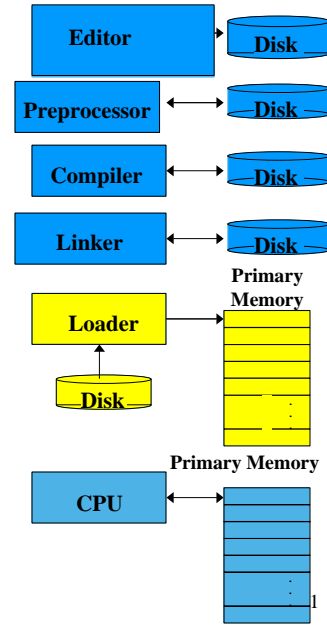


Loading (caricamento)



Prima che possa essere eseguito, un programma dovrà essere caricato nella memoria

Questa operazione è eseguita dal **loader** (caricatore) che prenderà il programma eseguibile dal disco e la trasferirà nella memoria



Esecuzione



Infine il computer eseguirà il programma, un'istruzione alla volta

Per caricare ed eseguire il programma in un sistema Linux si scrive

`./a.out`

La maggior parte dei programmi scritti in C ricevono e/o producono dei dati. Certe funzioni scritte in C prendono l'input (i dati di ingresso) dallo **stdin** (il dispositivo standard per l'input), assegnato normalmente alla tastiera

L'output (i dati di uscita) è inviato allo **stdout** (il dispositivo standard per l'output) che è normalmente lo schermo

Ma potrebbero essere connessi ad altri dispositivi!

C'è anche un dispositivo standard per l'errore **stderr** (normalmente è lo schermo)

