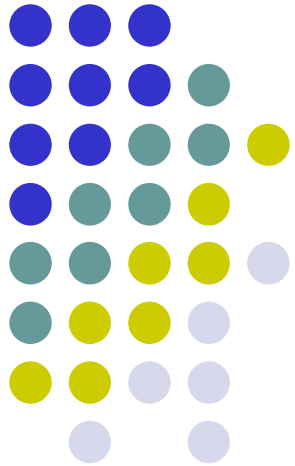
Laboratorio di Calcolatori 1

Corso di Laurea in Fisica

A.A. 2007/2008



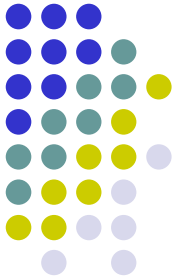
Dott.Davide Di Ruscio

Dipartimento di Informatica

Università degli Studi di L'Aquila

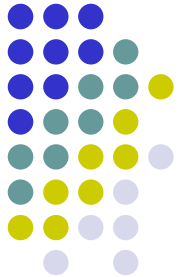
Lezione del 17/03/08

Nota



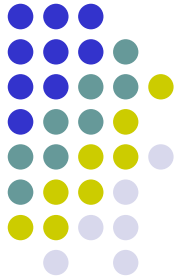
Questi lucidi sono tratti dal materiale distribuito dalla McGraw-Hill e basati su del materiale fornito dal Prof. Flammini Michele

Sommario (I parte)



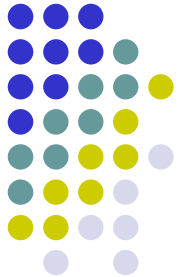
- Concetti fondamentali
- Aspetti architetturali di un sistema di calcolo
  - hardware
  - software
    - software di base
    - software applicativo
- Codifica dell'informazione
  - numeri naturali, interi, reali
  - caratteri
  - immagini
- Macchina di Von Neumann
  - CPU (UC, ALU, registri, clock)
  - memoria centrale
  - bus di bistema
  - periferiche
- Linguaggio macchina
- **Linguaggio assembler**
- **Sistema operativo**
- **Ambiente di programmazione**

Gestione delle Periferiche (1/4)



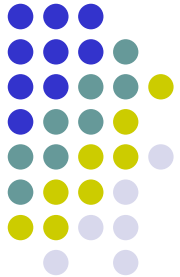
- I driver sono entità software cui è affidato il compito di comunicare dati da e verso le periferiche
- Essi garantiscono ai programmi che li usano una visione di alto livello (è possibile leggere o scrivere tramite primitive indipendenti dalla struttura hardware delle periferiche)
- Ad ogni operazione di alto livello corrisponde una serie di operazione di basso livello, gestite dal sistema operativo

Gestione delle Periferiche (2/4)



- I Driver si distinguono in
  - fisici (hardware)
    - per trasferire e manipolare
  - logici (software)
    - parte del sistema operativo che fornisce funzionalità ad alto livello che riguardano le periferiche

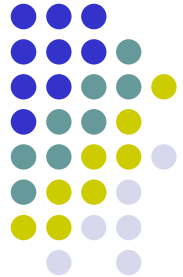
Gestione delle Periferiche (3/4)



Driver fisico

- Il device controller controlla i meccanismi fisici dell'apparecchiatura
  - (es. unità di lettura di floppy disk)
- Il device controller dialoga con la CPU attraverso registri e attraverso una memoria dedicata alle operazioni I/O chiamata DMA (Memoria ad accesso diretto)
- La DMA memorizza informazioni che il device controller può usare per scrivere in memoria direttamente senza passare attraverso la CPU

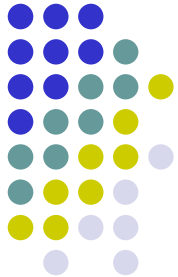
Gestione delle Periferiche (4/4)



Driver logico

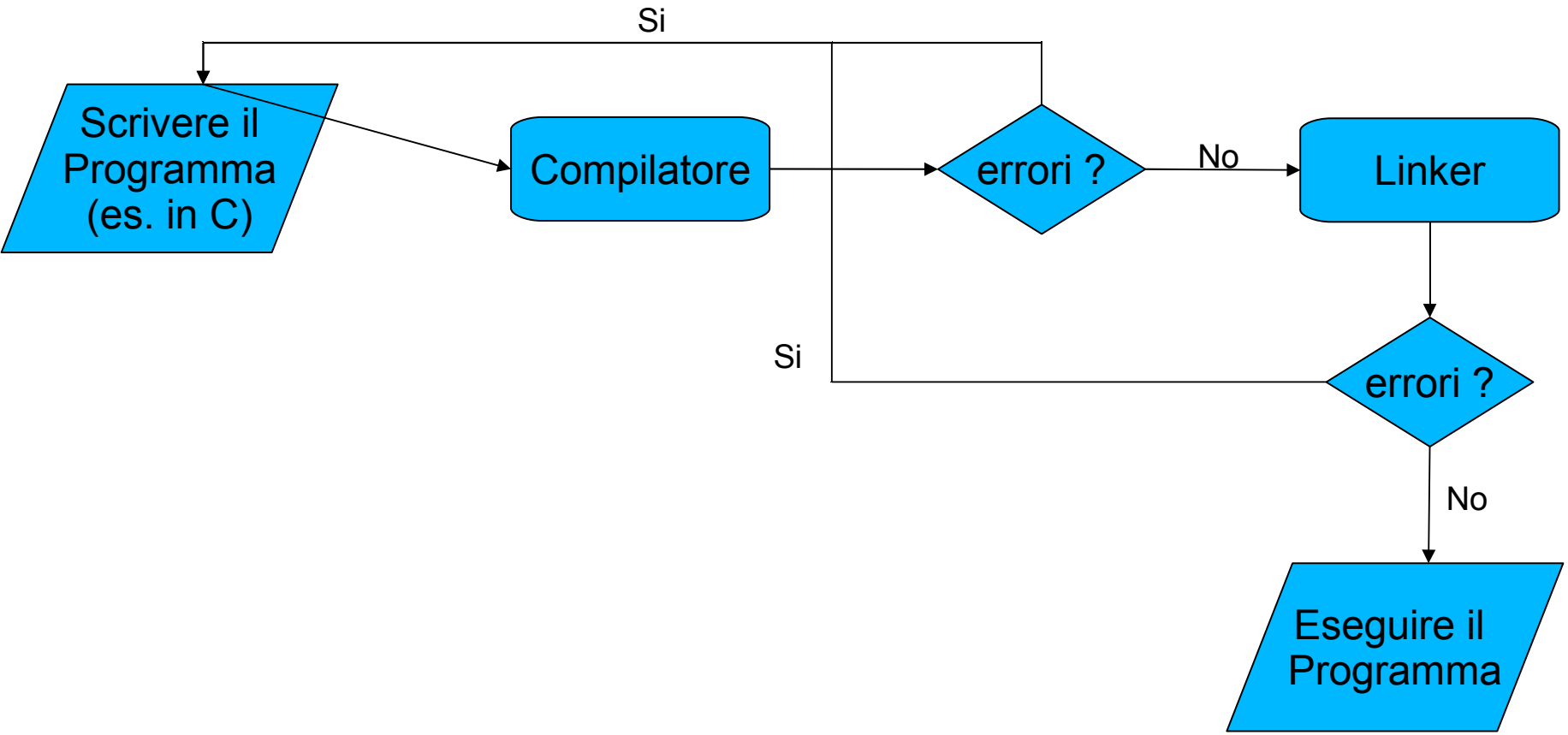
- Software che maschera i device fisici
  - Gestisce gli errori in lettura/scrittura
  - Gestisce i nomi del device driver
  - …

Ambiente di Programmazione

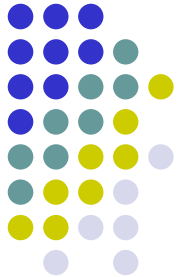


- Ogni linguaggio di programmazione è caratterizzato da un *insieme di strumenti che facilitano la scrittura dei programmi e la verifica della loro correttezza*
- Ogni ambiente di programmazione comprende:
  - Editor serve per scrivere programmi sorgente
  - Compilatore traduce un programma sorgente in un linguaggio direttamente eseguibile dall'elaboratore
  - Interprete esegue direttamente il codice sorgente senza tradurlo in linguaggio macchina
  - Linker permette di "collegare" insieme vari programmi (o moduli prodotti usando il compilatore) in un unico programma eseguibile
  - Debugger permette di controllare il programma durante la sua esecuzione

Il Processo di Programmazione



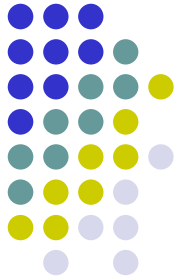
Sommario (II parte)



Il Linguaggio C

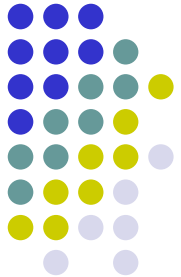
- Caratteristiche generali
- Un linguaggio C semplificato ed esempi di semplici programmi
- Struttura di un programma C
- Direttive del pre-processore
- Parte dichiarativa:
  - tipi
  - definizioni di tipi
  - definizioni di variabili
- Parte esecutiiva
  - istruzione di assegnamento
  - istruzioni (funzioni) di input-output
  - istruzioni di selezione
  - istruzioni iterative
- Vettori mono e multidimensionali
- Funzioni e procedure
- File
- Allocazione dinamica di memoria
- Suddivisione dei programmi in piu' file e compilazione separata
- Algoritmi elementari
  - ricerca sequenziale e binaria
  - ordinamento di un vettore: per selezione, per inserimento, per fusione e a bolle
- Aspetti avanzati di programmazione
  - ricorsione
  - strutture dati dinamiche

Il linguaggio C



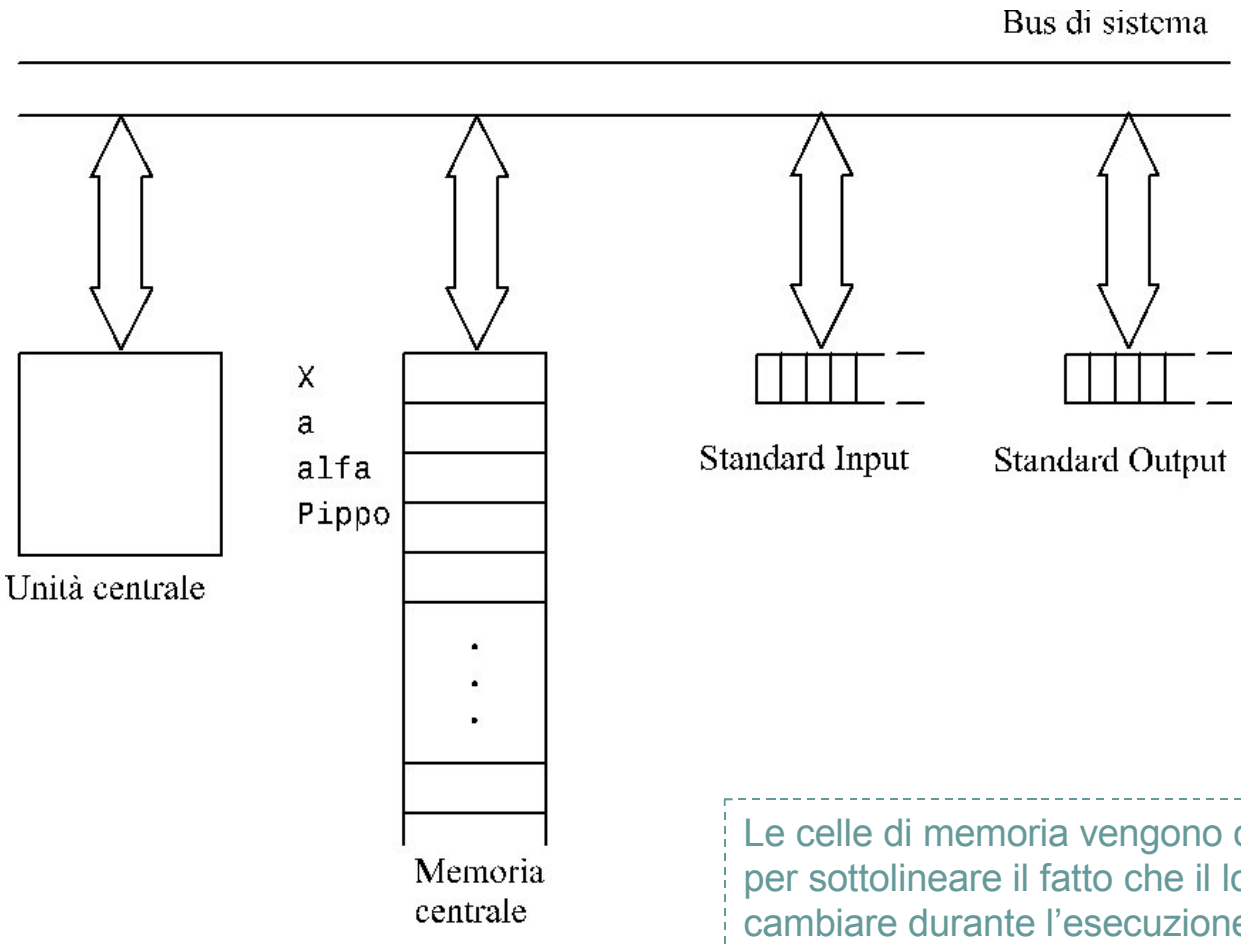
Caratteristiche generali

- Progettato da Dennis Ritchie nel 1972
- Sviluppato a partire dai linguaggi BCPL e B a cui aggiunge principalmente il concetto di tipo
- C è strettamente correlato con Unix che è stato scritto quasi completamente nel linguaggio C
- Il C si può collocare in una posizione intermedia fra un linguaggio a basso livello (assembler) ed uno ad alto livello.
- Possiede i principali costrutti di controllo propri di un linguaggio ad alto livello.



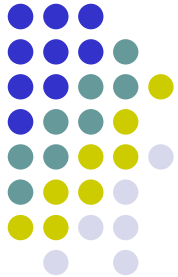
- Vedremo ora un linguaggio C semplificato ed alcuni semplici programmi
- Vedremo innanzitutto una macchina astratta su cui si appoggia l'esecuzione dei programmi
- Procederemo quindi con alcuni esempi di programmazione
- Per ogni esempio, tranne per i più semplici, mostreremo
  - l'idea risolutiva di base
  - lo pseudo-codice del programma, ossia una versione del programma in un formato discorsivo intermedio
  - il programma finale nel linguaggio C semplificato

La macchina astratta del C semplificato



Le celle di memoria vengono chiamate anche **variabili** per sottolineare il fatto che il loro contenuto può cambiare durante l'esecuzione del programma

I primi, semplici, programmi in "quasi" C

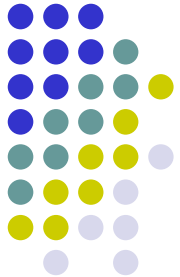


```
/*Programma NumeroMaggiore – prima versione */
main()
{
 scanf(x);
 scanf(y);
 if (x >  y) z = x; else z = y;
 printf(z);
}
```

- Le variabili, le istruzioni e altri elementi del programma sono indicati tramite **identificatori simbolici** definiti dal programmatore
- Un identificatore simbolico è definito come una successione di lettere e cifre, in cui al primo posto vi è una lettera
- Alcuni identificatori sono **predefiniti** e **riservati** in quanto associati a priori a qualche elemento del linguaggio e pertanto non possono essere usati dal programmatore con significati differenti da quello predefinito

```
/*Programma NumeroMaggiore –
                seconda versione */
main()
{
 scanf(x);
 scanf(y);
 if (x >  y) printf(x); else printf(y);
}
```

Struttura *sintattica* di un programma C

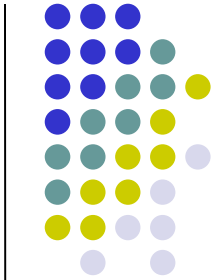


- Un programma C è composto da:
  - un'intestazione seguita da
  - una sequenza di istruzioni racchiusa tra i simboli **{** e **}**.

- L'**intestazione** è costituita *dall'identificatore* predefinito main seguito da una coppia di parentesi ( ) (per il momento vuote)

- Le **istruzioni** sono *frasi* del linguaggio di programmazione; ognuna di esse termina con il simbolo ';'.

Le principali istruzioni del quasi C



- Istruzione di assegnamento

```
x = 23;
w = 'a';
y = z;
r3 = (alfa*43–xgg)*(delta–32*ijj);
x = x+1;
```

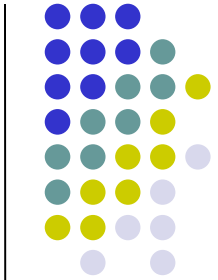
Identificatore '=' espressione

L'**espressione** può essere costituita da valori costanti, identificatori di variabili o una loro combinazione ottenuta mediante operatori aritmetici (+,-,*,/)

- Istruzioni di ingresso e uscita o I/O (scanf e printf)

```
scanf(x,y)
printf(y);
printf(x*y/z);
    temp=(x*y/z);
    printf(temp)
```

scanf e *printf* seguiti da una coppia di parentesi che racchiude l'identificatore di una variabile consentono, rispettivamente, di leggere e scrivere il valore di una variabile dallo Standard Input o sullo Standard Output



- Le istruzioni composte
  - Istruzione condizionale

Condizione

```
if(x == 0) z = 5; else y = z + w*y;
if(x == 0) {z = 5;} else {y = z + w*y;}
if ((x+y)*(z-2) > (23+v)) {z = x + 1; y = 13 + x;}
if ((x == y && z >3) || w != y) z = 5; else {y = z + w*y; x = z;}
```

Istruzioni scorrette:

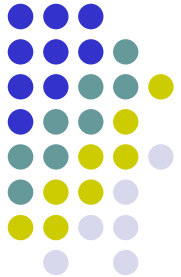
```
if (x == 0) else y = z; y = 34;
if (x == 0) a; else b + c;
```

Per **condizione** si intende un'espressione il cui valore può essere *vero* o *falso*. Essa è costruita mediante I normali operatori aritmetici, gli **operatori di relazione** (==,!=,<,>,<=,>=), e gli **operatori logici** (!,||, &&)

  - Istruzione iterativa o ciclica

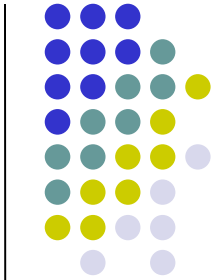
```
while (x >= 0) x = x – 1;
while (z != y) {y = z – x; x = x*3;}
```

Esempio 1



- Scrivere un programma in “quasi” C che legga numeri da input finché non viene inserito 0; appena viene inserito 0 il programma deve stampare 1.

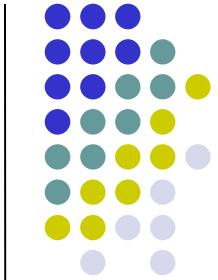
- Pseudocodice:
  - Leggi *dato*
  - Mentre *dato* è diverso da 0 leggi *dato*
  - Stampa 1



/*ProgrammaCercaIlPrimoZero */

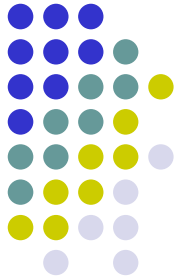
```
main()
{
 uno = 1;
 scanf (dato);
 while (dato !=0) scanf(dato);
 printf(uno);
}
```

Esempio 2

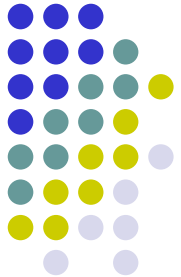


- Scrivere un programma in “quasi” C che legga numeri da input finché non viene inserito 0; appena viene inserito 0 il programma deve stampare la somma di tutti i numeri inseriti.

- Idea di risoluzione: manteniamo in una variabile *somma* la somma dei numeri inseriti fino al momento corrente



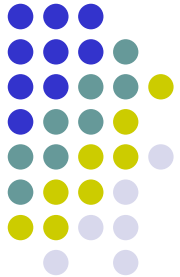
- Pseudocodice:
  - Inizializza *somma* a 0
  - Leggi *numero*
  - Mentre *numero* è diverso da 0 aggiungi *numero* a *somma* e leggi *numero* da input
  - Stampa *somma*



/*ProgrammaSommaSequenza */

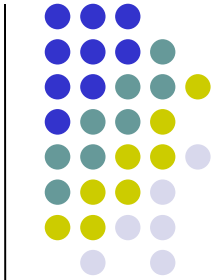
```
main()
{
 somma = 0;
 scanf(numero);
 while (numero != 0)
 {
     somma = somma + numero;
     scanf(numero);
 }
 printf(somma);
}
```

Esempio 3

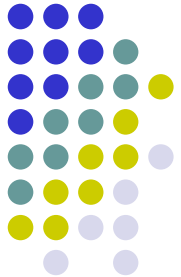


- Scrivere un programma in “quasi” C che legga numeri da input finché non viene inserito 0; appena viene inserito 0 il programma deve stampare il massimo di tutti i numeri inseriti.

- Idea di risoluzione:
  - manteniamo in una variabile *max* il massimo dei numeri inseriti fino al momento corrente
  - domanda: come bisogna inizializzare *max*?
  - si pone *max* pari al primo numero letto



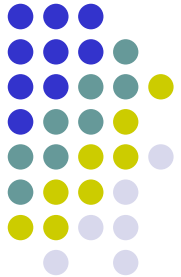
- Pseudocodice:
  - Leggi *numero*
  - Se *numero* è diverso da 0
    - poni *max* uguale a *numero*
    - leggi *numero*
    - mentre *numero* è diverso da 0
      - se *numero* è maggiore di *max* poni *max* uguale a *numero*
      - leggi *numero*
    - stampa *max*
  - Altrimenti stampa “sequenza vuota”



/*ProgrammaMaxSequenza */

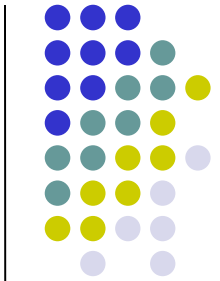
```
main()
{
 scanf(numero);
 if (numero != 0)
 {
      max = numero;
      scanf(numero);
      while (numero != 0)
      {
      if (numero > max) max = numero;
      scanf(numero);
      }
      printf(max);
 }
 else printf("Sequenza vuota");
}
```

Esempio 4

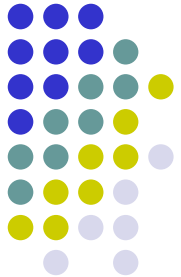


- Scrivere un programma in “quasi” C che stampa i numeri da 1 a n, dove il numero n è inserito in input.

- Idea di risoluzione:
  - effettuiamo un ciclo tramite l’utilizzo di un contatore inizializzato a 1 e incrementato ogni volta di 1 fino a raggiungere n
  - in ogni iterazione del ciclo stampiamo il valore del contatore



- Pseudocodice:
  - Leggi *n*
  - Inizializza *i* ad 1
  - Mentre *i* è minore o uguale a *n* stampa i e incrementa *i* di 1



```
main()
{
 scanf(n);
 i=1;
 while (i<=n)
 {
    printf(i);
    i=i+1;
 }
}
```