

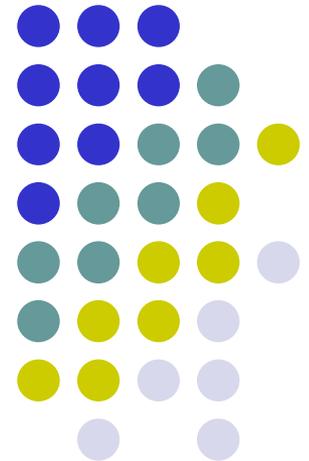
# Laboratorio di Calcolatori 1

Corso di Laurea in Fisica

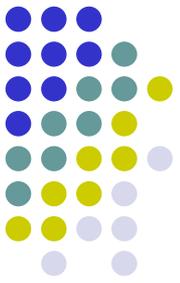
A.A. 2007/2008

Dott. Davide Di Ruscio

Dipartimento di Informatica  
Università degli Studi di L'Aquila



# 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 esecutiva
  - 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

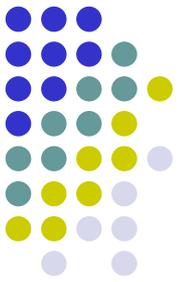
- Algoritmi elementari
  - ricerca sequenziale e binaria
  - ordinamento di un vettore: per selezione, per inserimento, per fusione e a bolle
- Aspetti avanzati di programmazione
  - ricorsione
  - strutture dati dinamiche

## RIFERIMENTI

Ceri, Mandrioli, Sbattella  
[Informatica arte e mestiere](#)  
McGraw-Hill

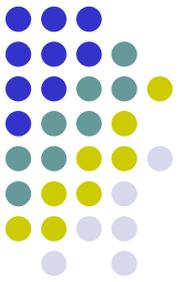
**N.B. (copyright): alcuni di questi lucidi sono tratti dal materiale distribuito dalla McGraw-Hill**

# Nota

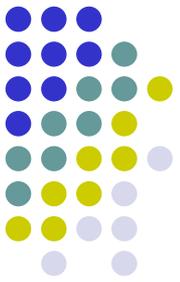


Questi lucidi sono tratti dal materiale distribuito dalla McGraw-Hill e basati su del materiale fornito dal Prof. Flammini Michele

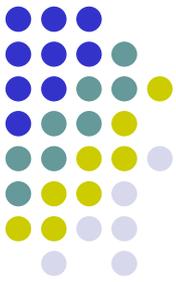
## Esempio 3



- 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



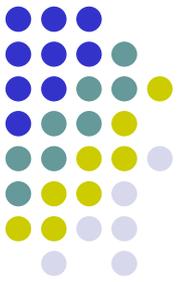
- Pseudocodice:
  - Leggi *numero*
  - Se *numero* è diverso da 0
    - poni *max* uguale a *numero*
    - leggi *numero*
    - mentre *numero* è diverso da 0
      - se *numero* è maggiore di *max* poni *max* uguale a *numero*
      - leggi *numero*
    - stampa *max*
  - Altrimenti stampa “sequenza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



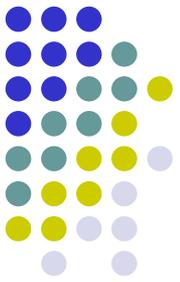
```
/*ProgrammaMaxSequenza */
```

```
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");
}
```

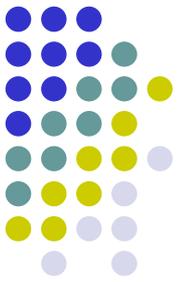
## Esempio 4



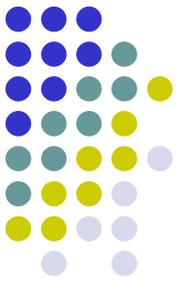
- Scrivere un programma in “quasi” C che stampa i numeri da 1 a n, dove il numero n è inserito in input.
- Idea di risoluzione:
  - effettuiamo un ciclo tramite l'utilizzo di un contatore inizializzato a 1 e incrementato ogni volta di 1 fino a raggiungere n
  - in ogni iterazione del ciclo stampiamo il valore del contatore



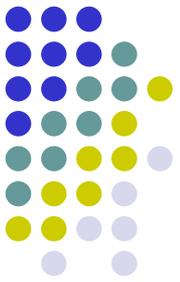
- Pseudocodice:
  - Leggi  $n$
  - Inizializza  $i$  ad 1
  - Mentre  $i$  è minore o uguale a  $n$  stampa  $i$  e incrementa  $i$  di 1



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

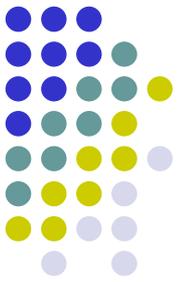


- Vediamo ora una soluzione alternativa con l'istruzione ciclica **for**
- Pseudocodice:
  - Leggi  $n$
  - Per  $i$  che varia da  $1$  a  $n$  incrementando  $i$  di  $1$  ad ogni passo stampa  $i$

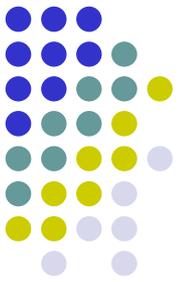


```
main()  
{  
  scanf(n);  
  for (i=1; i<=n; i=i+1)  
    printf(i);  
}
```

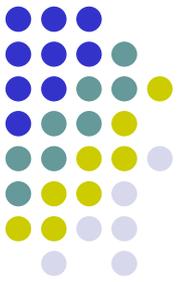
## Esempio 5



- Scrivere un programma in “quasi” C che stampa i numeri da  $n$  a 1, dove il numero  $n$  è inserito in input.
- Idea di risoluzione:
  - effettuiamo un ciclo tramite l'utilizzo di un contatore inizializzato a  $n$  e decrementato ogni volta di 1 fino a raggiungere 1
  - in ogni iterazione del ciclo stampiamo il valore del contatore



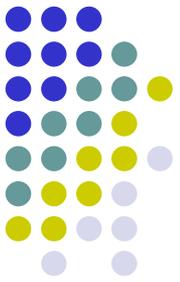
- Pseudocodice:
  - Leggi  $n$
  - Per  $i$  che varia da  $n$  a  $1$  decrementando  $i$  di  $1$  ad ogni passo stampa  $i$



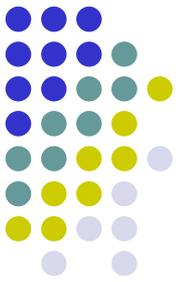
```
main()  
{  
  scanf(n);  
  for (i=n; i>=1; i=i-1)  
    printf(i);  
}
```

**Esercizio:** riscrivere il programma utilizzando l'istruzione **while** al posto di quella **for**

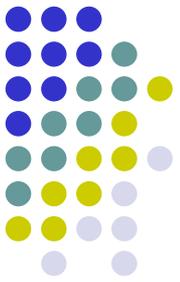
## Esempio 6



- Scrivere un programma in “quasi” C che lette in input le lunghezze dei tre lati di un triangolo verifica se il triangolo è valido e di che tipo di triangolo si tratta.
- Idea di risoluzione:
  - Effettuiamo dei controlli sulle lunghezze dei lati per assicurarci che il triangolo sia valido e per individuare le sue proprietà (isoscele, equilatero, scaleno)



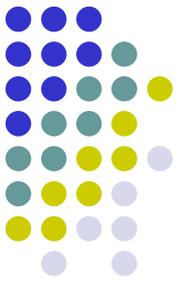
- Pseudocodice:
  - Leggi le lunghezze dei tre lati
  - Se nessun lato è maggiore della somma degli altri due
    - Se i tre lati sono uguali
      - stampa “Il triangolo è equilatero”
    - altrimenti
      - se due lati sono uguali
        - stampa “Il triangolo è isoscele”
      - altrimenti
        - stampa “Il triangolo è scaleno”
  - altrimenti
    - stampa “Il triangolo non è valido”



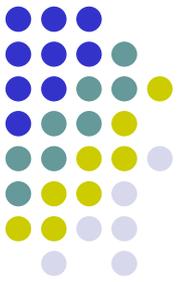
`/*Programma per la valutazione di un triangolo */`

```
main()
{
    /*Lettura dei dati di ingresso */
scanf(X); scanf(Y); scanf(Z);
    /* Verifica che i dati possano essere le lunghezze
    dei lati di un triangolo */
if ((X < Y + Z) && (Y < X + Z) && (Z < X + Y))
    /*Distinzione tra i vari tipi di triangolo */
    if (X == Y && Y == Z)
printf("I dati letti corrispondono a un triangolo equilatero");
    else
    if (X == Y || Y == Z || X == Z)
printf("I dati letti corrispondono a un triangolo isoscele");
    else
printf("I dati letti corrispondono a un triangolo scaleno");
else
printf("I dati letti non corrispondono ad alcun triangolo");
}
```

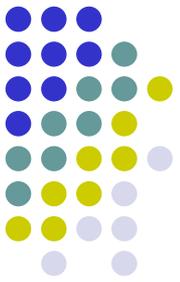
## Esempio 7



- Scrivere un programma in “quasi” C che calcoli il massimo di  $n$  numeri letti in input, dove il numero  $n$  è inserito in input.
- Idea di risoluzione:
  - manteniamo in una variabile *max* il massimo dei numeri inseriti fino al momento corrente

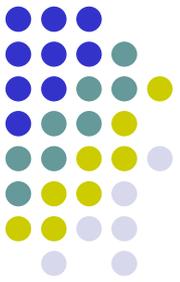


- Pseudocodice:
  - Leggi  $n$
  - Se  $n$  è maggiore di 0
    - Leggi  $dato$
    - Inizializza  $max$  a  $dato$
    - Per  $i$  che varia da 2 ad  $n$  incrementando  $i$  di 1 ad ogni passo
      - Leggi  $dato$
      - memorizza in  $max$  il massimo tra  $max$  e  $dato$
    - Stampa  $max$
  - Altrimenti stampa “Sequenza vuota”

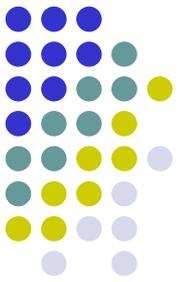


```
main()
{
  scanf(n);
  if (n > 0)
  {
    scanf(dato);
    max = dato;
    for (i=2; i<=n; i=i+1)
    {
      scanf(dato);
      if (dato>max) max=dato;
    }
    printf(max);
  }
  else printf("Sequenza vuota");
}
```

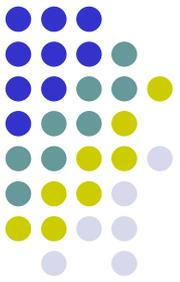
## Esempio 8



- Scrivere un programma in “quasi” C che legga in input un numero  $n$  e stampi il suo fattoriale.
- Idea di risoluzione:
  - Calcoliamo tramite un ciclo il prodotto dei numeri da 2 a  $n$  (se  $n$  è minore di 2 restituiamo 1)
  - Ciò viene ottenuto ponendo una variabile  $f$  inizialmente pari ad 1 e quindi moltiplicandola per tutti i numeri da 2 ad  $n$

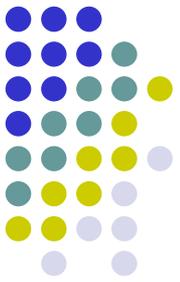


- Pseudocodice:
  - Leggi  $n$
  - Inizializza  $f$  a  $1$
  - Per  $i$  che varia da  $2$  ad  $n$  incrementando  $i$  di  $1$  ad ogni passo
    - moltiplica  $f$  per  $i$ , ossia poni  $f$  uguale al prodotto  $f * i$
    - incrementa  $i$  di  $1$
  - Stampa  $f$

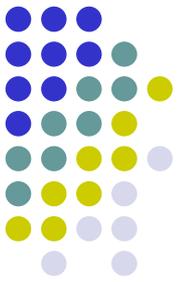


```
main()  
{  
  scanf(n);  
  f=1;  
  for (i=2; i<=n; i=i+1)  
    f=f*i;  
  printf (f);  
}
```

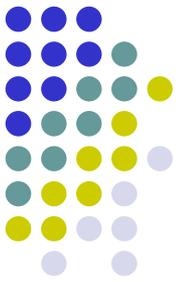
## Esempio 9



- Scrivere un programma in “quasi” C che legga in input due interi  $a$  e  $b$  e stampi il risultato della potenza  $a^b$ .
- Idea di risoluzione:
  - $a^b$  è pari ad  $a*a*...*a$   $b$  volte
  - Tramite un ciclo moltiplichiamo quindi una variabile  $p$  inizialmente posta ad  $1$  per  $a$   $b$  volte.

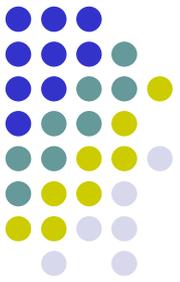


- Pseudocodice:
  - Leggi  $a$ ,  $b$
  - Inizializza  $p$  a 1
  - Mentre  $b$  è maggiore di 0
    - moltiplica  $p$  per  $a$
    - decrementa  $b$  di 1
  - Stampa  $p$



```
main ()
{
  scanf (a);
  scanf (b);
  p=1;
  while (b>0)
  {
    p=p*a;
    b=b-1;
  }
  printf (p);
}
```

# Nota sull'istruzione if



```
if (C1) if (C2) S1; else S2;
```

significa:

```
if (C1)  
    if (C2) S1;  
else S2;
```

oppure

```
if (C1)  
    if (C2) S1;  
    else S2;
```

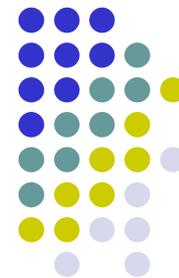
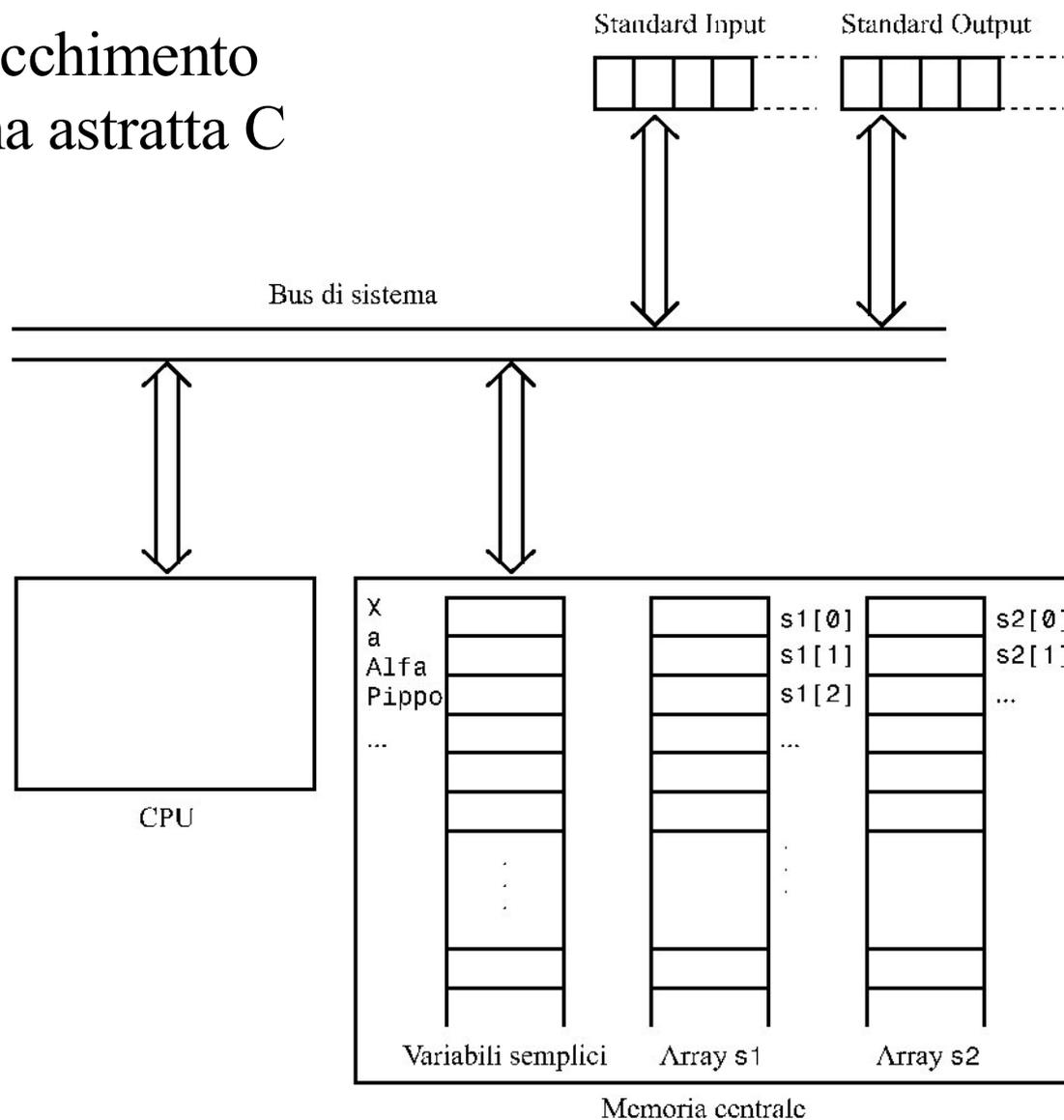
?

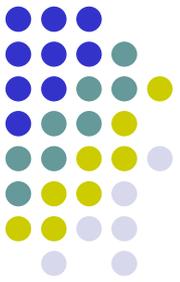
**Convenzione:** il primo ramo **else** viene attribuito all'ultimo **if**. Altrimenti, scriviamo esplicitamente

```
if (C1) {if (C2) S1;} else S2;
```

# Le variabili strutturate

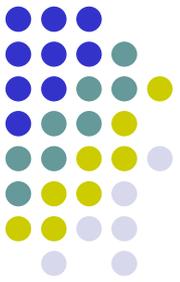
- Un primo arricchimento della macchina astratta C





# I vettori (array) (1/2)

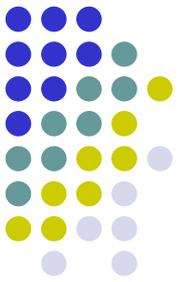
- Per array si intende una sequenza di celle di memoria consecutive ed omogenee, cioè contenenti tutte dati tra loro uniformi
- Un array viene identificato come qualsiasi altra variabile
- Però anche i suoi elementi sono variabili
- Ad ognuna di queste sequenze viene dato un unico nome o identificatore
- La singola cella di una sequenza viene identificata mediante il nome della sequenza e un indice



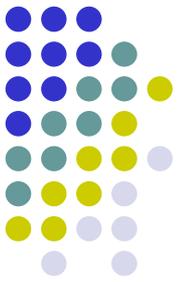
# I vettori (array) (2/2)

- Esempi:
  - `scanf(s[2]);`
  - `a[3] = s[1] + x;`
  - `if (a[4] > s[1] + 3) s[2] = a[2] + a[1];`
  - `x = a[i];`
  - `a[i] = a[i+1];`  
`a[i*x] = s[a[j+1]-3]*(y - a[y]);`
- In C il primo elemento di ogni array è sempre lo 0-esimo

# Esempio 10

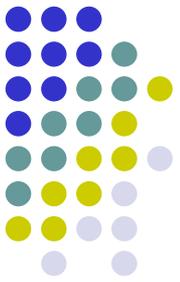


- Scriviamo un programma in “quasi” C che legga in input una sequenza di caratteri terminante con ‘%’ e stampi la sequenza in ordine inverso rispetto all’ordine di inserimento
- Idea di risoluzione:
  - Memorizziamo i valori della sequenza immessa in un array fino all’inserimento di ‘%’. Stampiamo gli elementi dell’array a partire dall’ultimo fino ad arrivare al primo.



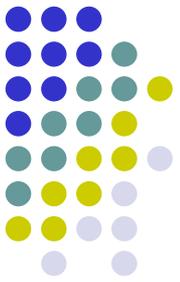
- Pseudocodice:
  - Inizializza  $i$  a 0
  - Leggi  $x$
  - Mentre  $x$  è diverso da ‘%’
    - Memorizza  $x$  in posizione  $i$  nell’array *sequenza*
    - incrementa  $i$  di 1
    - Leggi  $x$
  - Decrementa  $i$
  - Mentre  $i$  è maggiore o uguale a 0
    - Stampa il contenuto della posizione  $i$  dell’array *sequenza*
    - Decrementa  $i$  di 1.

/\* Programma InvertiSequenza \*/

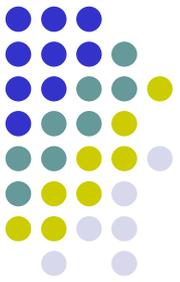


```
main()
{
    i = 0;
    scanf(x);
    while (x != '%')
    {
        sequenza[i] = x;
        i = i + 1;
        scanf(x);
    }
    i=i-1;
    while (i >= 0)
    {
        printf(sequenza[i]);
        i = i - 1;
    }
}
```

# Esempio 11

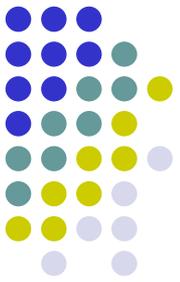


- Scrivere un programma in “quasi” C che determini l’indice del minimo elemento di un vettore di  $n$  numeri letto in input.
- Idea di risoluzione:
  - Scandendo in sequenza le componenti a partire dalla prima, manteniamo in una variabile *imin* l’indice dell’elemento minimo finora letto.



- Pseudocodice:

- Leggi  $n$
- Se  $n$  è maggiore di 0
  - Per  $i$  che varia da 0 ad  $n-1$ 
    - leggi l'elemento di indice  $i$  del vettore
  - Inizializza  $imin$  a 0
  - Per  $i$  che varia da 1 ad  $n-1$  incrementando  $i$  di 1 ad ogni passo
    - Se l'elemento di indice  $i$  è minore di quello di indice  $imin$  poni  $imin$  uguale a  $i$
  - Stampa  $imin$
- Altrimenti stampa "Vettore vuoto"



```
main()
{
  scanf(n);
  if (n > 0)
  {
    for (i=0; i<n; i=i+1)
      scanf(a[i]);
    imin = 0;
    for (i=1; i<n; i=i+1)
      if (a[i]<a[imin]) imin=i;
    printf(imin);
  }
  else printf("Vettore vuoto");
}
```