i Vettori

Alessandra Giordani
agiordani@disi.unitn.it
Lunedì 15 aprile 2013
http://disi.unitn.it/~agiordani/

Dichiarazione di un Array

- La dichiarazione di un array di n elementi di tipo T causa l'allocazione contigua di **n variabili di tipo T**. Ad es:

```
int myarr[7];
```

- dichiara un array di 7 interi chiamato myarr.
- Il nome dell'array (`myarr`) è un puntatore alla prima variabile / cella di memoria allocata
  - L'indirizzo del 2° elemento dell'array e dato da `myarr + 1`
  - L'indirizzo del 3° elemento dell'array e dato da `myarr + 2`
  - … e così via, ma lo vedremo meglio più avanti!
- Attenzione: il 1° elemento dell'array ha indice 0, non 1!

Array e Puntatori

- In C gli array sono intrinsecamente legati al concetto di **puntatore**
- Per capire veramente come funzionano gli array (e per sfruttarli a dovere) bisogna capire i puntatori
- L'utilizzo dei puntatori (e dell'allocazione dinamica della memoria) consentono un uso molto più flessibile degli array
- MA LO VEDREMO PIU' AVANTI QUANTO PARLEREMO DI PUNTATORI

Accesso ad elementi di array

Le parentesi quadre [*n*] si utilizzano:

- Per dichiarare un array di un *n* elementi
  - int a[10]; *//n deve essere un intero!*
- Per accedere al'*n*-esimo elemento dell'array
  - Sia in lettura (se a dx di un = o in printf)
  - Sia in scrittura (se a sx di un = o in scanf)
  - a[5] = a[5] + 1;

Inizializzazione di un vettore

- Valori indefiniti in alcuni elementi dell'array possono provocare errori difficili da rilevare
- Occorre inizializzare il vettore:
  - Dichiarare l'array static (gli elementi del vettore non inizializzati vengono posti a zero)
  - Valori diversi possono essere specificati, facendoli seguire alla dichiarazione dell'array, racchiusi fra parentesi graffe

Copia del vettore a in vettore b

```c
#include <stdio.h>
#define MAX 5

int main(){
int a[MAX] = {11, -2, -63, 4.5, 15};
static int b[MAX], i;
  //a = b; //ERRORE!
  printf("Valori di b prima:\n");
  for (i = 0; i < MAX; i++)
      printf("%d ",b[i]);
  for (i = 0; i < MAX; i++)
      b[i] = a[i];
  printf("\n Valori di b dopo:\n");
  for (i = 0; i < MAX; i++)
      printf("%d ",b[i]);
  printf("\n");
}
```

- Non si può fare `a=b`
  - Provate!
- Se dichiariamo b senza static..
  - Cosa stampa?
- Cosa succede al valore 4.5 assegnato 4° elemento di a?

Lettura di un vettore da stdin

```c
#include <stdio.h>
#define MAX 5

int main()
{
    int a[MAX];
    int i,tmp;

    for (i=0;i<MAX;i++)
    {
        printf("Inserire il %d° numero: ", i+1);
        scanf("%d",&tmp);
        a[i]=tmp;
    }
    printf("\nContenuto del vettore:\n");
    for (i=0;i<MAX;i++)
        printf("[%d]",a[i]);
    printf("\n");
    return 0;
}
```

- Mediante scanf()
- Interpreta gli spazi e gli acapo come separatori

//o equivalentemente `scanf("%d",&a[i]);`

Dati n numeri trova il minimo

- Non è essenziale usare i vettori per implementare tale programma
  - Avevamo visto lo stesso esempio con i flow chart
  - Provare a farlo per casa…
- Ma vediamo quale sarebbe il programma che trova il minimo in un vettore di numeri
  - Il vettore è inizializzato in fase di dichiarazione
  - Si potrebbe leggere i dati da stdin con scanf()
  - O se fossero digit [0..9] con getchar

Dati n numeri trova il minimo

- Non occorre dichiarare la dimensione tra []
- Ma specificando la dimensione n e un numero di elementi <n i rimanenti elementi
  - verranno inizializzati a 0 se il vettore è dichiarato static
  - conterrano valori a caso, altrimenti

```
#include <stdio.h>
#define MAX 5
int main()
{
    float min,a[]={3,4.5,10,2,9};
    int i;

    min=a[0];
    for (i=1;i<MAX;i++)
    {
        if (min>a[i])
          min=a[i];
    }
    printf("Min = %f\n",min);
    return 0;
}
```

Inverti una lista di numeri

- Supponiamo siano digit [0..9]
  - usiamo getchar()
  - conversione già vista sfruttando ascii
- Possiamo implementarlo SENZA usare i vettori?
  - NO perché dobbiamo leggerli e memorizzarli tutti per poi stamparli in ordine inverso
  - MENTRE il minimo lo calcolavamo mano a mano che li leggevamo

Inversione di un vettore

- Leggo *dim* numeri (dim<10)
- Li inserisco via via nel vettore *a*
- Inserisco in *b* gli elementi di *a* in ordine inverso
- Stampo vettori

PARTIAMO DA 0

```c
#include <stdio.h>
#define MAX 10

int main()
{
    int a[MAX], b[MAX];
    int i, dim=5;
    for(i=0;i<dim;i++)
    {
        printf("Inserire l'elemento a[%d]: ",i+1);
        scanf("%d", &a[i]);
    }
    /*stampa vettore*/
    printf("\nIl vettore e':\n\n");
    for(i=0;i<dim;i++)
        printf("a[%d] = %d\n", i, a[i]);
    /*inversione vettore*/
    for(i=0;i<dim;i++)
        b[i]=a[dim-1-i];
    /*stampa vettore invertito*/
    printf("Il vettore invertito e':\n\n");
    for(i=0;i<dim;i++)
        printf("b[%d] = %d\n", i, b[i]);
    return 0;
}
```

Esercizio: leggi in vettore e trova il massimo

- Scrivere un main che
  - legga un vettore di interi
  - lo memorizzi correttamente
  - stampi il vettore letto
  - stampi il valore massimo

Input di stringhe

- Queste considerazioni sono particolarmente delicate nel caso in cui vogliamo usare scanf per leggere stringhe
- La sequenza %s all'interno di un template consuma qualunque sequenza di caratteri che non contenga spazi
- Per leggere una stringa, dobbiamo prima avere allocato un buffer (cioe un array di caratteri) abbastanza capiente. Ad esempio:

```
char buf [256]; // crea un buffer
scanf("%s", buf);
```

- Se la stringa immessa dall'utente è piu lunga del buffer (nell'esempio la lunghezza massima è 255, tenendo conto del null character '\0') abbiamo un errore di **segmentation fault** (violazione di accesso in memoria)