ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Meccanica

Elementi di Informatica

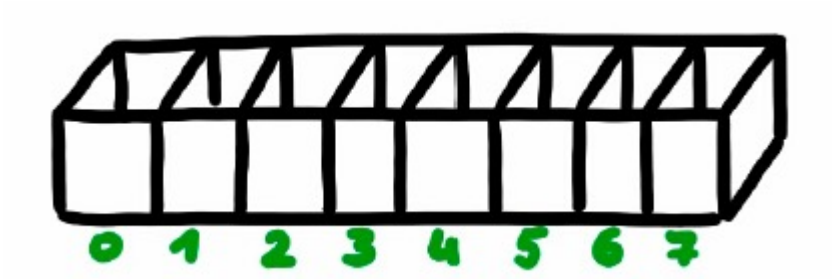
Array – Parte II

Giordano Da Lozzo e Giuseppe Sansonetti

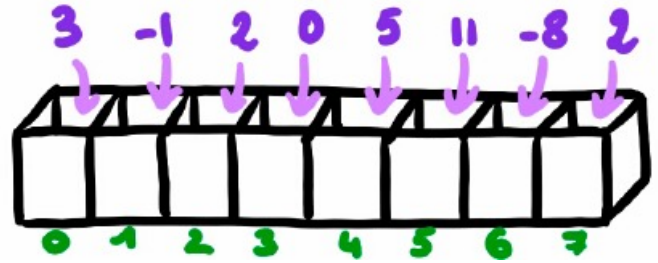
Previously… Array

E’ una sequenza di variabili omogenee.

Permette di accedere in lettura ed in scrittura a ciascuna delle variabili che lo compongono sulla base della loro posizione, detta indice.



Previously… Array



Array composto da
8 variabili di tipo int

E’ una sequenza finita di variabili dello stesso tipo.

Elementi: le variabili che compongono un array prendono il nome di elementi dell’array.

Sequenza: gli elementi dell’array sono ordinati, quindi si può parlare del primo elemento, del secondo elemento, …

Sequenza finita: il numero di variabili che compongono l’array è un numero finito, detto lunghezza o dimensione dell’array.

Indice: numero intero non negativo associato a ciascun elemento, ne indica la posizione. Le posizioni sono 0, 1, 2, …, lunghezza-1.

Previously… Array: concetti ulteriori

Gli array (monodimensionali) sono anche detti vettori.

Gli array costituiscono una struttura dati, ovvero un'entità o un metodo utilizzato per memorizzare insiemi di dati.

- Il metodo per memorizzare dati associato ad un array è quello di memorizzare i valori degli elementi uno dopo l'altro (logicamente e vedremo anche fisicamente), nell'ordine in cui compaiono nell'array.

Gli array sono variabili strutturate, mentre le variabili semplici sono non strutturate.

Gli array hanno un tipo, che è il tipo dei loro elementi.

Previously… Dichiarazione di array

Per usare un array, è necessario dichiarare una variabile che permetta di referenziare l'array, specificando tipo, nome e lunghezza dell'array.

Esempi: int interi[20]; float reali[15/3];

Sintassi: tipo nome[lunghezza];

Semantica: alloca un'area di memoria sufficiente a contenere l'array.

Nota: lunghezza deve essere una espressione di tipo int

```
int interi[3.5];
```

```
error: size of array 'interi' has non-integer type
```

Previously… Accesso

Dopo aver dichiarato una variabile per referenziare l'array, è possibile utilizzare l'array.

- L'unico utilizzo che si può fare di un array è accedere ai suoi elementi per memorizzare un valore al loro interno (accesso in scrittura) o per recuperare un valore già memorizzato al loro interno (accesso in lettura).

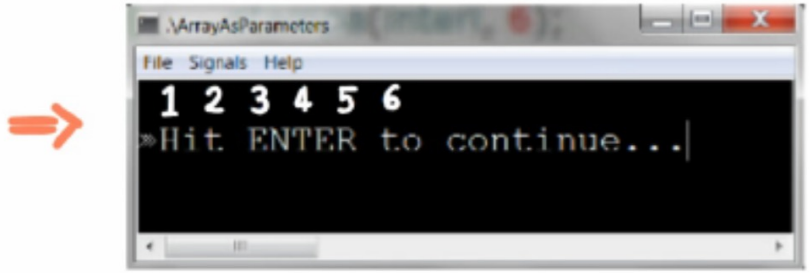
Sintassi: *nome*[*indice*]

Semantica: accede all'elemento con indice *indice* all'interno dell'array referenziato da *nome*.

Array e parametri

E’ possibile che un parametro di una funzione sia un array?

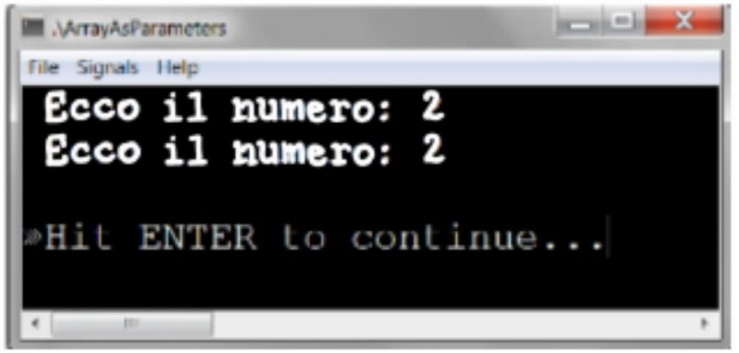
```
void stampa(int array[], int size) {
     for(int i=0; i<size; i++)
          printf("%d ", array[i]);
}

int main() {
     int interi[6] ={1,2,3,4,5,6};
     stampa(interi, 6);
}
```



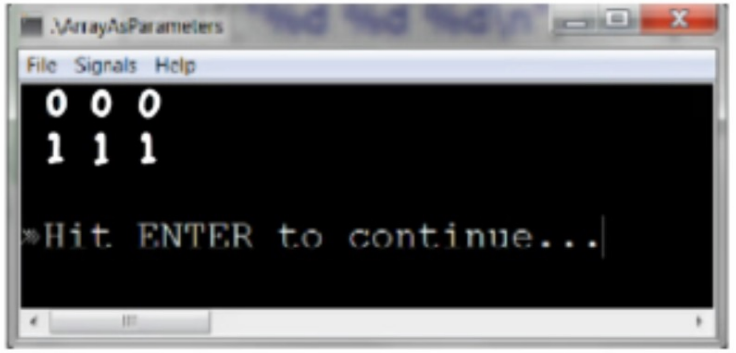
Tutto sembra funzionare ma…

Qualcosa non va?

```
void modificaValore(int numero) {
    numero = numero + 1;
}

int main() {
    int intero = 2;
    printf(”Ecco il numero: %d\n", intero);
    modificaValore(intero);
    printf(”Ecco il numero: %d\n", intero);
}
```




```
void incrementaElementi(int interi[], int lunghezza) {
    for(int i=0; i<lunghezza; i++)
        interi[i]++;
}

int main() {
    int num[3] ={0,0,0};
    printf("%d %d %d\n", num[0], num[1], num[2]);
    incrementaElementi(num, 3);
    printf("%d %d %d\n", num[0], num[1], num[2]);
}
```



Array come parametri

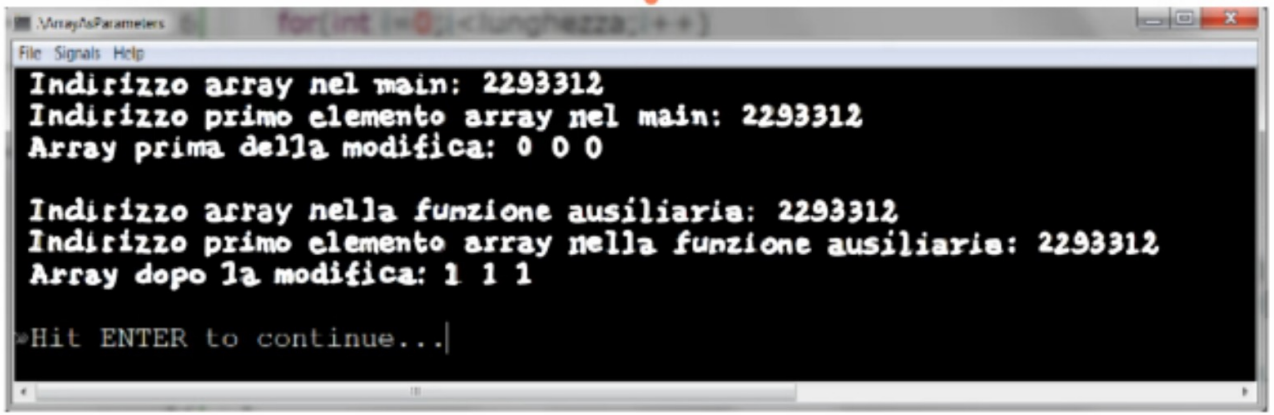
Il meccanismo di legame fra parametri di tipo primitivo si chiama per valore: il valore del parametro attuale viene copiato nel parametro formale, che viene utilizzato come una variabile indipendente dal parametro attuale.

- Gli array in C non possono essere passati ad una funzione per valore.

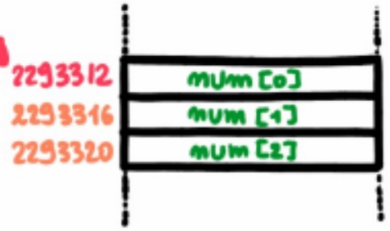
- Gli array in C vengono passati per riferimento (ovvero viene passato l'indirizzo di memoria del primo elemento dell'array).
  - In realtà, viene passato per valore l'indirizzo del primo elemento dell'array, ma possiamo pensare che sia passato direttamente l'indirizzo.

Passaggio dei parametri per riferimento

```c
void incrementaElementi(int interi[], int lunghezza) {
     printf("Indirizzo array nella funzione ausiliaria: %d\n", interi);
     printf("Indirizzo primo elemento array nella funzione ausiliaria: %d\n", &interi[0]);
     for(int i=0;i<lunghezza;i++)
          interi[i]++;
}

int main() {
     int num[3] ={0,0,0};
     printf("Indirizzo array nel main: %d\n", num);
     printf("Indirizzo primo elemento array nel main: %d\n", &num[0]);
     printf("Array prima della modifica: %d %d %d\n\n", num[0], num[1], num[2]);
     incrementaElementi(num, 3);
     printf("Array dopo la modifica: %d %d %d\n", num[0], num[1], num[2]);
}
```



CIÒ CHE VIENE EFFETTIVAMENTE PASSATO DALLA FUNZIONE *main* ALLA FUNZIONE *incrementaElementi* È L'INDIRIZZO DEL PRIMO ELEMENTO DELL'ARRAY.



Funzione invocante

Per passare un riferimento ad un array (ovvero l'indirizzo del primo elemento dell'array) ad una funzione è sufficiente scrivere il nome dell'array, in quanto tale nome è associato all'indirizzo del primo elemento dell'array.

`incrementaElementi(num, 3);` invoca la funzione *incrementaElementi* fornendogli come parametro l'indirizzo del primo elemento dell'array.

- Si può infatti anche scrivere `incrementaElementi(&num[0], 3);`

Funzione invocata

La funzione invocata riceve un indirizzo – il fatto che riceva un array è puramente concettuale – e memorizza tale indirizzo in una variabile.

```
void incrementaElementi(int interi[], int lunghezza) {
```

L'indirizzo ricevuto come parametro attuale viene salvato nel parametro formale *interi*.

- Adesso *interi* può essere usato nella funzione invocata come ogni altro array, in particolare *interi[i]* permette di accedere all'elemento il cui indirizzo di memoria è pari a quello di *interi* più *4*i*, dove *4* è la dimensione di un elemento di tipo *int*.

Attenzione: le modifiche realizzate dalla funzione invocata sono avvertite anche nella funzione invocante.

Quali parametri formali?

Opzione 1: *funzione(tipo nome[], int lunghezza)*

Nota: non è possibile ricavare la lunghezza di un array, per questo è comune che la lunghezza venga richiesta come un ulteriore parametro.

Opzione 2: *funzione(tipo *nome, int lunghezza)*

Nota: il parametro è un puntatore. Nelle opzioni 1 e 2 il parametro formale *nome* «decade» (viene trasformato) in un puntatore.

- E' per semplicità concettuale che viene ammesso che un parametro formale possa avere la stessa forma di un array dichiarato localmente.

Restituire un array?

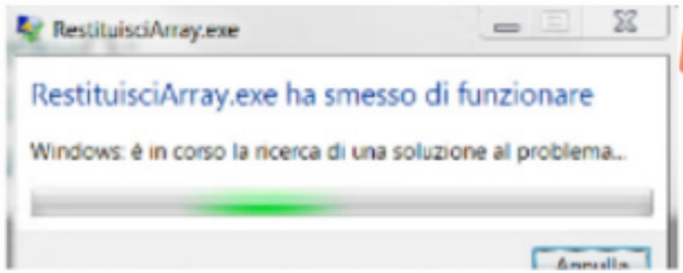
In C le funzioni non possono restituire un array!

```
int[] creaArray() {
    int sequenza[10] ={1,2,3,4,5,6,7,8,9,10};
    return sequenza;
}

int main(){
    int array[10] = creaArray();
    for(int i=0; i<10; i++)
        printf("%d ", array[i]);
}
```

⇒ error: expected identifier or '(' before '[' token

Le funzioni possono restituire un puntatore ma comunque non si può utilizzare un puntatore ad un array statico che è stato creato in una funzione la cui esecuzione è terminata, in quanto quando una funzione termina, le sue variabili locali vengono deallocate dalla memoria

```
int* creaArray() {
    int sequenza[10] ={1,2,3,4,5,6,7,8,9,10};
    return sequenza;
}

int main(){
    int* puntatore;
    puntatore= creaArray();
    for(int i=0; i<10; i++)
        printf("%d ", puntatore[i]);
}
```



warning: function returns address of local variable [-Wreturn-local-addr]

Somma degli elementi in un array

SOMMA ELEMENTI

Realizzare un programma che consiste di due funzioni.

La funzione main legge una sequenza di interi introdotta dall'utente (prima di leggere tale sequenza chiede all'utente di quanti numeri consiste la sequenza) ed invoca la funzione sommaElementi per calcolare la somma degli interi introdotti dall'utente. Stampa quindi un messaggio che informa l'utente di tale somma.

Useremo spesso questa imprecisione!

La funzione *sommaElementi* riceve come parametro un array di interi, la sua lunghezza e restituisce la somma degli interi nell'array.

```c
#include <stdio.h>

/* funzione che riceve come parametro un array di interi e restituisce la somma degli interi nell'array */
int sommaElementi(int interi[], int dimensione) {
     // pre: nessuna
     int somma = 0; // somma degli elementi, inizialmente zero
     /* somma tutti gli elementi alla stessa variabile */
     for(int i=0; i<dimensione; i++)
          somma += interi[i];
     return somma;
}

/* programma che legge una sequenza di interi introdotta dall'utente e stampa la somma di tali interi. */
int main() {
     int lunghezza; // lunghezza della sequenza
     int somma;

     /* leggi la lunghezza della sequenza */
     printf(”Caro utente, quanti numeri hai intenzione di introdurre?\n");
     scanf("%d", &lunghezza);

     /* leggi la sequenza */
     int sequenza[lunghezza];

     for(int i=0; i<lunghezza; i++) {
          printf(”Caro utente, introduci un numero.\n");
          scanf("%d", &sequenza[i]);
          }

     /* calcola e stampa il risultato */
     somma = sommaElementi(sequenza, lunghezza);
     printf("La somma degli elementi vale %d", somma);
}
```

Triplica gli elementi di una sequenza

TRIPLICA ELEMENTI

Realizzare un programma che consiste di due funzioni.

La funzione main legge una sequenza di interi introdotta dall'utente (prima di leggere tale sequenza chiede all'utente di quanti numeri consiste la sequenza) ed invoca la funzione triplicaElementi per triplicare il valore degli elementi della sequenza. La nuova sequenza viene quindi stampata.

la funzione *triplicaElementi* riceve come parametro un array di interi, la sua lunghezza e triplica il valore dei suoi elementi.

```c
#include <stdio.h>
/* funzione che riceve come parametro un array di interi e ne triplica il valore degli elementi */
void triplicaElementi(int interi[], int dimensione) {
     // pre: nessuna
     /* triplica il valore di ciascun elemento */
     for(int i=0; i<dimensione; i++)
        interi[i] *= 3;
}

/* programma che legge una sequenza di interi introdotta dall'utente e stampa la sequenza ottenuta triplicando il
valore degli elementi  della sequenza letta. */
int main() {
          int lunghezza; // lunghezza della sequenza

     /* leggi la lunghezza della sequenza */
     printf("Caro utente, quanti numeri hai intenzione di introdurre?\n");
     scanf("%d", &lunghezza);

     /* leggi la sequenza */
     int sequenza[lunghezza];
     for(int i=0; i<lunghezza; i++) {
          printf("Caro utente, introduci un numero.\n");
          scanf("%d", &sequenza[i]);
     }

     /* triplica il valore degli elementi */
     triplicaElementi(sequenza, lunghezza);
     printf("Ecco la nuova sequenza: ");
     for(int i=0; i<lunghezza; i++)
          printf("%d ", sequenza[i]);
}
```

Altre risorse

- Bellini, Guidi: Linguaggio C — 10.1 – 10.5