

A decorative vertical bar on the left side of the slide, consisting of several vertical lines of varying shades of blue and grey. To the right of these lines are several blue circles of different sizes, some overlapping the lines. The largest circle is at the top left, and several smaller ones are scattered below it.

Laboratorio di Programmazione 1

**Docente: dr Damiano Macedonio
Lezione 13 - 23/01/2014**

1

Funzioni e Array

- Oltre a variabili di tipo primitivo e valori, è possibile passare come argomento di una funzione il valore di un elemento di un array, oppure un intero array.
- Un elemento di un'array con tipo base primitivo è trattato come una qualsiasi variabile di tipo primitivo: il suo valore è *copiato* all'interno del parametro formale.
- Il passaggio di un'array come argomento di una funzione ha un funzionamento diverso.

Funzioni e Array: Dichiarazione

- Dichiarazione di una funzione che ha un parametro di tipo array:
 - `int minimum (int values[]);`
- Non è necessario indicare nella dichiarazione della funzione la dimensione dell'array: il compilatore ignora questa informazione.
- Per far funzionare la funzione con array di dimensione diversa è possibile aggiungere un parametro che contiene tale informazione:
 - `int minimum (int values[], int size);`

Funzioni e Array: Comportamento

- Quando viene passato un array come argomento di una funzione, la funzione riceve l'indirizzo di memoria in cui si trova l'array. Riferendosi a tale indirizzo, la funzione ha accesso proprio all'array originale.
- *Se una funzione che riceve un array come argomento, cambia il valore di un elemento di tale array, tale modifica viene apportata all'array originale che è stato passato alla funzione.*

Funzioni e Array: Comportamento

```
int sum (int a, int b);  
int main (void) {  
    ...  
    s = sum(x, y);  
    ...  
}
```

- Il valore delle variabili x e y passate come argomenti sono copiati nei parametri formali a e b . Qualsiasi modifica fatta ad a e b non ha effetto su x e y .
- Questo vale anche se al posto di x e y si usano due elementi di un array:

```
s = sum(a[0], a[1]);
```

Funzioni e Array: Comportamento

```
int sum (int x[]) {  
    x[2] = x[0] + x[1];  
}
```

```
int main(void) {  
    int a[3] = {4, 6, 3};  
    sum (a);  
    ...  
}
```

- L' *indirizzo della memoria* contenente l'array *a* è copiato nel parametro formale *x*. Qualsiasi modifica fatta al contenuto di tale locazione di memoria dalla funzione *sum* persiste anche dopo la terminazione di tale funzione.

Funzioni e Array Multidimensionali

- Analogamente a quanto accade per gli array monodimensionali, è possibile:
 - Passare *un elemento* di un array multidimensionale ad una funzione.
 - Si comporta come una qualsiasi variabile “semplice”.
 - Passare un array multidimensionale ad una funzione.
 - Come per gli array monodimensionali, qualsiasi modifica apportata al contenuto dell’array ha effetto anche dopo l’esecuzione della funzione.

Funzioni e Array Multidimensionali: Dichiarazione

- Nella dichiarazione di una funzione che ha come argomento un array bi-dimensionale è possibile omettere nella dichiarazione numero di righe, *ma non il numero di colonne*:
 - `int sum(int array_values[10][5])` VALIDO
 - `int sum(int array_values[][5])` VALIDO
 - `int sum(int array_values[10][])` NON VALIDO
 - `int sum(int array_values[][])` NON VALIDO
- `int array_values[10][5]`: è un array di 10 elementi ciascuno dei quali ha tipo array di 5 elementi interi.
- `int array_values[][5]`: è un array di x elementi ciascuno dei quali ha tipo array di 5 elementi interi.
- `int array_values[10][]`: è un array di 10 elementi ciascuno dei quali ha un *tipo non noto*!
 - Il compilatore deve sapere il tipo degli elementi per garantire la corretta invocazione delle funzioni!