PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Fondamenti di programmazione

Docente: Catalano Rocco

Argomento: Funzioni

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

Funzioni

- Con il termine *funzione* si intende, in generale, un operatore che, applicato a un insieme di operandi, consente di calcolare un risultato, come avviene anche per una funzione matematica f(x).

- f(x) restituisce un valore, in generale, diverso per ogni diverso valore di x. I valori che inseriamo tra parentesi, che in matematica chiamiamo *variabili indipendenti*, in C si chiamano *parametri della funzione*

Funzioni

- In C, per *definire una funzione*, occorre prima *dichiarare* il cosiddetto *prototipo* della funzione nella sezione delle dichiarazioni globali (fuori dal main(), insieme alle variabili che possono essere viste da tutti i blocchi di codice che costituiscono il programma):

```
<tipo risultato> <nome funzione> (<elenco
   parametri>);
```

dove:

```
<elenco parametri> ::=
                <tipo  parametro>  [<nome  parametro>]
              {,<tipo parametro> [<nome parametro>]}
```

Funzioni

- Dichiarare una funzione permette al compilatore di fare un controllo sul tipo dei parametri che le verranno poi passati quando verrà usata

Esempi:
```
int somma (int m, int n);
int somma (int, int);
int fun(void);
```

- void è un tipo di dato speciale che rappresenta *assenza di valori:* quindi fun è una funzione che restituisce un valore intero ma non in dipendenza di un *parametro*, ma di ciò che avviene al suo interno durante l'esecuzione (es. lettura di un dato).

NB Non è obbligatorio dare un nome ai parametri nella dichiarazione di una funzione: basta il tipo. Le due dichiarazioni della funzione somma sono entrambe valide

Funzioni

- Occorre poi *definire* la funzione

  <tipo risultato> <nome funzione> ([<elenco par.>])

- All'interno del corpo della funzione si utilizza un'apposita istruzione per terminare l'esecuzione della funzione e restituirne il risultato:

  `return` *<espressione risultato>*

Esempio

Es.

```
#include<stdio.h>

int doppio(int);  /* dichiarazione funzione */

int doppio(int x){  /* definizione funzione */
return ( 2 * x );
}

 int main(){     /* programma principale che */
 int d, p=5;     /* usa la funzione */
 d = doppio(p);
 printf("%d", d);
 return 0;    /* anche main() è una funzione */
}
```

Funzioni con parametri

- Possono operare su valori diversi per ogni *chiamata di funzione*, a seconda del valore che assumono in quel momento le espressioni che sono inserite nella lista dei parametri

- Si dice che i parametri vengono passati *per valore*

Passaggio parametri per valore

- **Parametri formali** utilizzati nella definizione della funzione: indicano *come* operare sui valori che vengono passati nella chiamata ma 'non esistono' al di fuori della funzione.
- La funzione che utilizza al suo interno un'altra funzione si dice, rispetto ad essa, *funzione chiamante;* l'utilizzo di una funzione si dice *chiamata di funzione.*
- La funzione opera *sui valori senza alterare la variabile* eventualmente utilizzata nella chiamata.
- Ogni chiamata deve passare tanti valori, attraverso una serie di espressioni dello stesso tipo dei parametri corrispondenti, quanti sono i parametri formali.

Passaggio di parametri per valore

- **Parametri effettivi** passati per valore dalla *funzione chiamante*
- I valori dei dati effettivi sono copiati nei parametri formali utilizzati dalla funzione chiamata
- Nessun effetto provocato da modifiche nel parametro formale all'interno della funzione si ripercuote sul parametro reale del programma chiamante

Esempio

```
#include<stdio.h>
int doppio(int);

int main() {
int g = 5, h;
h = doppio(g);
printf("%d %d", g, h);
return 0;
}
int doppio(int x) /* x è un parametro formale */
{return ( 2 * x );}
```

Quale sarà il valore stampato per le variabili g e h ?

Funzioni: il passaggio degli array

- Gli array non sono passati per valore
- Gli array vengono passati *per indirizzo*
- Quindi la funzione lavora realmente sulle variabili che si trovano a quell'indirizzo
- Se la funzione modifica il contenuto dell'array, tale modifica si riflette sull'array originario
- Non occorre specificare la dimensione dell'array nell'elenco dei parametri formali