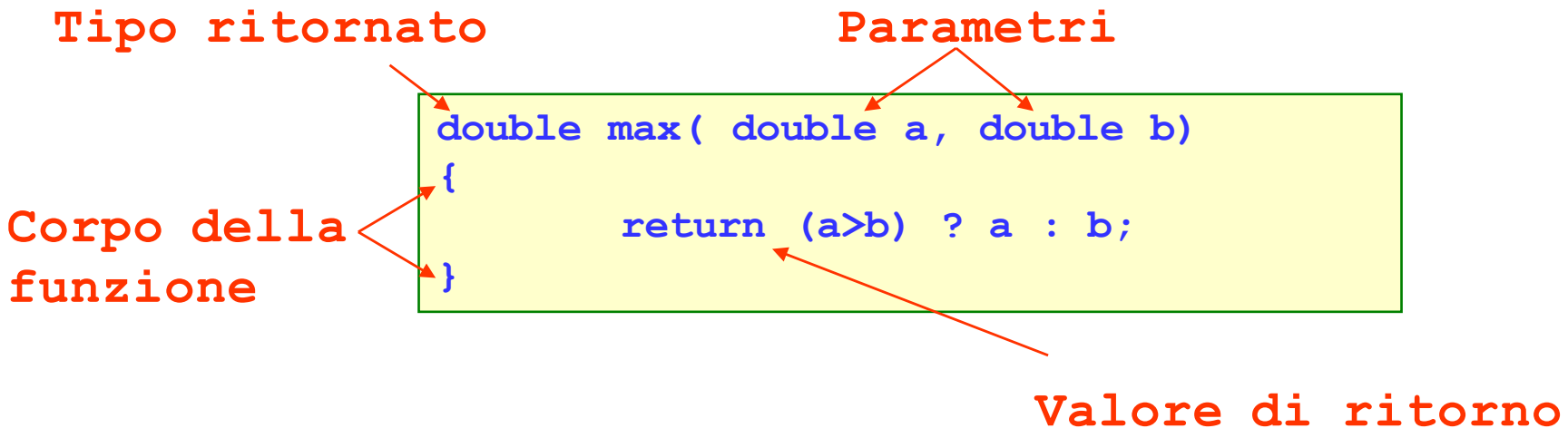
Ambienti di Programmazione per il Software di Base

- Le Funzioni in C
- Esercizi sulle Funzioni svolti
- Esercizi sulle Funzioni da svolgere

Funzioni

- In C le funzioni sono caratterizzate da un nome, dal tipo della variabile ritornata e da una lista di parametri (opzionali)



- La lista dei parametri (anche se vuota) deve essere esplicitata
- Il valore ritornato deve essere compatibile, a meno di conversione esplicita, con il tipo della funzione

Prototipi delle funzioni

- Prima di essere usata, una funzione deve essere dichiarata (nel file che la usa)

main.c

```
#include <stdio.h>
double max(double, double);
int main()
{
        double m = max(1, 3);
        printf("Il massimo e` %d\n", m );
        return 0;
}
```

max.c

```
double max (double a, double b)
{
        return (a>b) ? a : b;
}
```

Prototipo di max (normalmente in **max.h**)

- I prototipi rendono le funzioni in C "*type safe*", nel senso che i valori reali degli argomenti vengono all'occorrenza convertiti nei tipi formali specificati dal prototipo

Chiamata per valore

- Con il passaggio per valore la funzione non può modificare il valore dei suoi argomenti

```
bool greater(int i, int j) { // se i>j scambia i e j
        if (i>j) {
                int temp=i;
                i=j;
                j=temp;
                return true;
        }
        else
                return false;
}
```

Argomenti passati "per valore" sono scambiati nella funzione ma nel programma chiamante hanno i valori vecchi

- **Es: int i = 5, j=7;**
- **Dopo la chiamata di greater (i,j)**
- **i vale sempre 5 e j vale sempre 7**

Puntatori come parametri

- L'uso dei puntatori permette ad una funzione di modificare il valore dei suoi argomenti

```
bool greater(int *p1, int *p2) { /* se il valore
puntato da p1 > valore puntato da p2 scambia i due
valori*/
        if (*p1>*p2) {
                int temp=*p1;
                *p1=*p2;
                *p2=temp;
                return true;
        }
        else
                return false;
}
```

Se passo i puntatori le modifiche Hanno effetto sul programma chiamante

  - **Es: int i = 5, j=7;**
  - **Dopo la chiamata di greater (&i,&j)**
  - **i varrà 7 e j 5**

Chiamata per riferimento

- L'uso dei riferimenti permette ad una funzione di modificare il valore dei suoi argomenti

```
bool greater(int& i, int& j) { // se i>j scambia i e j
        if (i>j) {
                int temp=i;
                i=j;
                j=temp;
                return true;
        }
        else
                return false;
}
```

Argomenti passati "by reference" possono essere modificati dalla funzione stessa

  - Con i riferimenti è sufficiente chiamare greater (i, j);
  - Per ragioni di efficenza, oggetti di grandi dimensioni (in termini di memoria) vengono normalmente passati "*by reference*".

Nota : La chiamata per riferimento non esiste in C ma esiste in C++, per compilare codice C++ usare il comando g++ al posto di gcc

Esercizi

1. Scrivere una funzione che riceva come parametro un puntatore ad un vettore, e restituisce:
   Vero, se la somma dei numeri pari contenuti nel vettore è maggiore della somma dei numeri dispari.
   Falso altrimenti.

Consiglio: (Oltre che il puntatore, è buona norma che il metodo riceva anche la dimensione del vettore).

2. Scrivere una funzione *alternaVettori* che riceve due vettori, e restituisce un terzo vettore costruito interponendo gli elementi dei primi due (i due vettori hanno uguale dimensione).

3. Scrivere una funzione per il calcolo del fattoriale di N.

4. Scrivere due funzioni *trovaMax* che trovino il massimo e la posizione del massimo in vettori di double.

Esercizi

5. Scrivere un programma per la gestione di matrici di double, che contenga i seguenti metodi:

Una funzione create, che riceve due interi (dimensione di righe e colonne), crea una matrice delle dimensioni indicate, e restituisce il suo puntatore.
Un metodo *erase*, che azzera tutti gli elementi della matrice.
Un metodo *read* per la lettura di una matrice da tastiera.
Un metodo *print* per la stampa della matrice su output.
Un metodo *mult* che effettui il prodotto di due matrici, restituendo il puntatore alla matrice risultante.