



Corso di Programmazione a oggetti

**Overloading delle funzioni e
parametri di default**

a.a. 2014/2015

Francesco Fontanella

Overloading delle funzioni

- In C++, è possibile dare a funzioni diverse lo stesso nome, a condizione che le funzioni abbiano liste di parametri diverse (in numero e/o tipo).
- Il compilatore è in grado di associare in modo univoco ciascuna chiamata a una delle funzioni, distinte in base alla lista degli argomenti

■ Esempio

Pensiamo di voler definire delle funzioni di incremento di puntatori per diversi possibili casi. Volendo però usare lo stesso nome.

```
// A: incrementa di uno la variabile di tipo int
void incr(int &x) {
    x++;
}
```

```
// B: incrementa di uno una variabile di tipo float
void incr(float &x) {
    x = x + 1.0;
}
```

SEGUE...

```
// C: incrementa di dx la variabile di tipo int
void incr(int &x, int dx) {
    x += dx;
}

main () {

float a=0.0;
int i = 1;
int d = 10;

incr(a); // versione B: si incrementa x (float)

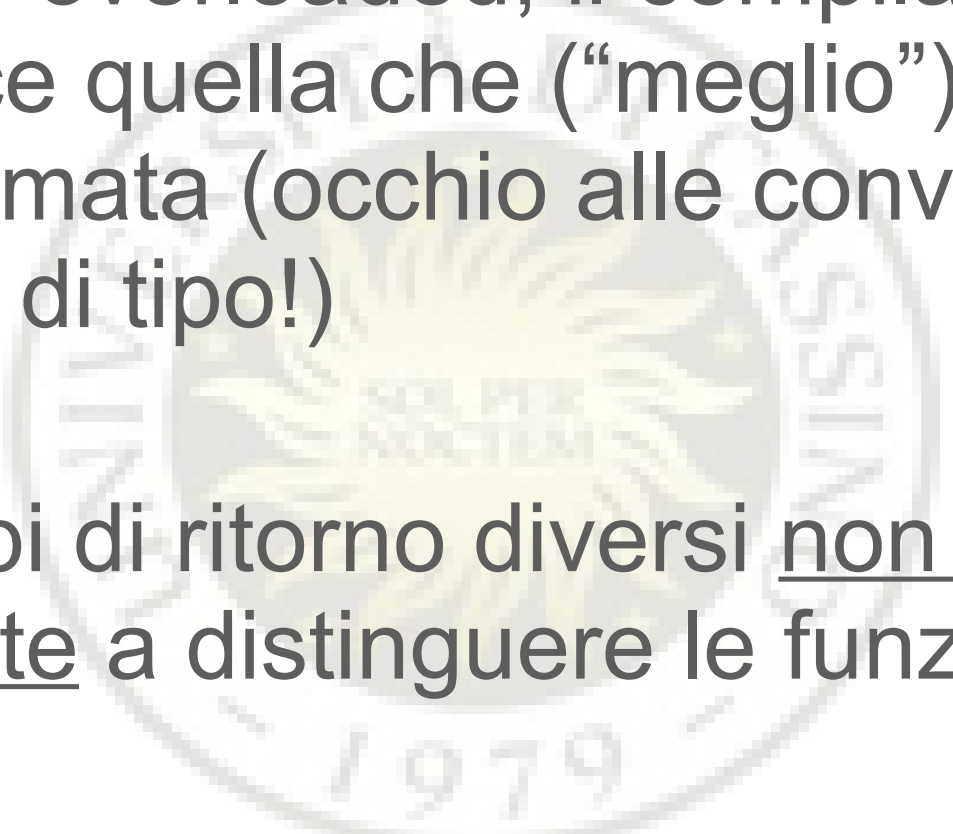
incr(i); // versione A: si incrementa i (int)

incr(i, d); // versione C: si incrementa i di d

}
```

La firma delle funzioni

- Quando uno stesso nome corrisponde a più funzioni, si dice che tale nome è “sovraccaricato” (overloaded) di significato.
- La cosiddetta Firma (signature) di una funzione è costituita da:
nome + tipo dei parametri

- 
- In presenza della chiamata di una funzione overloaded, il compilatore riconosce quella che (“meglio”) si adatta alla chiamata (occhio alle conversioni implicite di tipo!)
 - Avere tipi di ritorno diversi non è sufficiente a distinguere le funzioni
 - Nell’istruzione di chiamata infatti può non essere evidente il tipo del valore di ritorno.

Overloading di funzioni e ambiguità

- Esistono situazioni in cui il compilatore non è in grado di distinguere tra due o più funzioni in overloading. In questo caso si parla di ambiguità. Le ambiguità generano errori di compilazione.
- Le situazioni ambigue sono di solito causate dalle conversioni automatiche di tipo.

Esempio

```
double myfunc(double i); // versione A  
float myfunc(float i); // versione B
```

```
int main()  
{  
  
    myfunc(10.1);  
    myfunc(10);  
  
    return 0;  
}
```

NON AMBIGUA: chiama la A.
le costanti con virgola mobile
sono automaticamente di tipo
double

AMBIGUA: come deve essere
convertita la costante?
In float o double?

Parametri di default



Parametri di default

- In C++ è possibile specificare dei valori default per i parametri.
- Questi parametri possono essere gli ultimi (oppure *tutti*).
- **IMPORTANTE!:** E' il prototipo che deve specificare gli argomenti di default.
- Se l'invocazione della funzione non specifica gli ultimi argomenti, questi vengono assunti uguali agli argomenti di default.

```
void myfunc(int a, int b, int c =0);
```

```
int main() {  
  int x,y;
```

```
  myfunc(x,y);
```

```
}
```

La funzione prende 3 parametri,
il terzo è omissso in quanto è
quello di default (che vale 0)

```
void PrintValues(int val1, int val2=10)
{
    cout << "1st value: " << nValue1 << endl;
    cout << "2nd value: " << nValue2 << endl;
}

int main()
{
    PrintValues(1); // secondo parametro: default (è 10)
    PrintValues(3, 4); // secondo parametro: è 4

    return;
}
```

OUTPUT

```
1st value: 1
2nd value: 10
1st value: 3
2nd value: 4
```

- Una funzione può avere tutti i parametri di default:

```
void PrintValues(int val1=10, int val2=10, int val3=30)
{
    cout<<"Values: "<<val1<<" "<<val2<<" "<<val3<<" "<<endl;
}
```

```
int main()
{
    PrintValues(1, 2, 3);
    PrintValues(1, 2);
    PrintValues(1);
    PrintValues();
}
```

OUTPUT

```
Values: 1 2 3
Values: 1 2 30
Values: 1 20 30
Values: 10 20 30
```

- Per fornire un valore diverso da quello di default per `val3` è necessario fornire anche i valori di default per `val1` e `val2`

```
PrintValues(10, 20, 3);
```

- Infatti la chiamata:

```
PrintValues(3);
```

è equivalente a

```
PrintValues(3, 20, 30);
```

- I parametri di default devono stare per ultimi:

void PrintValue(**int** val1=10, **int** val2);

vietato!

- I parametri di default più a sinistra dovrebbero essere quelli che più di frequente sono diversi da quelli di default.