Alessio Bechini
- Corso di Fondamenti di Informatica II -

Generalizzazione di funzioni e di classi

Macro come funzioni "generiche"


```
int intMax(int x, int y) {
   return (x>y) ? x : y;
}
```

```
long longMax(long x, long y) {
   return (x>y) ? x : y;
}
```

```
double doubleMax(double x, double y) {
   return (x>y) ? x : y;
}
```

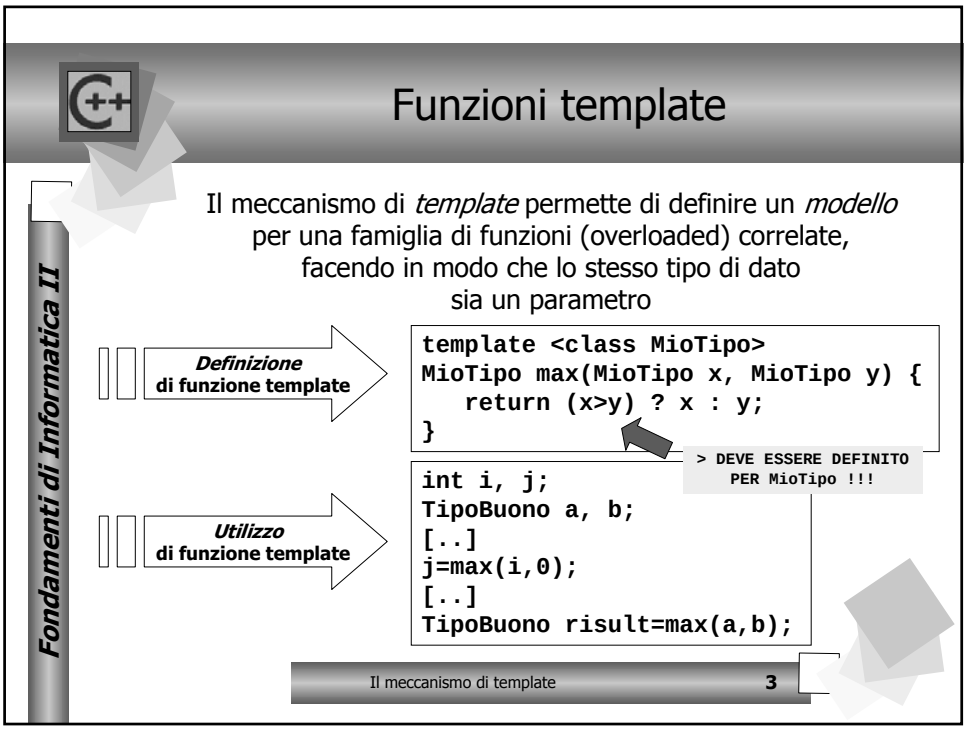
Possibile soluzione

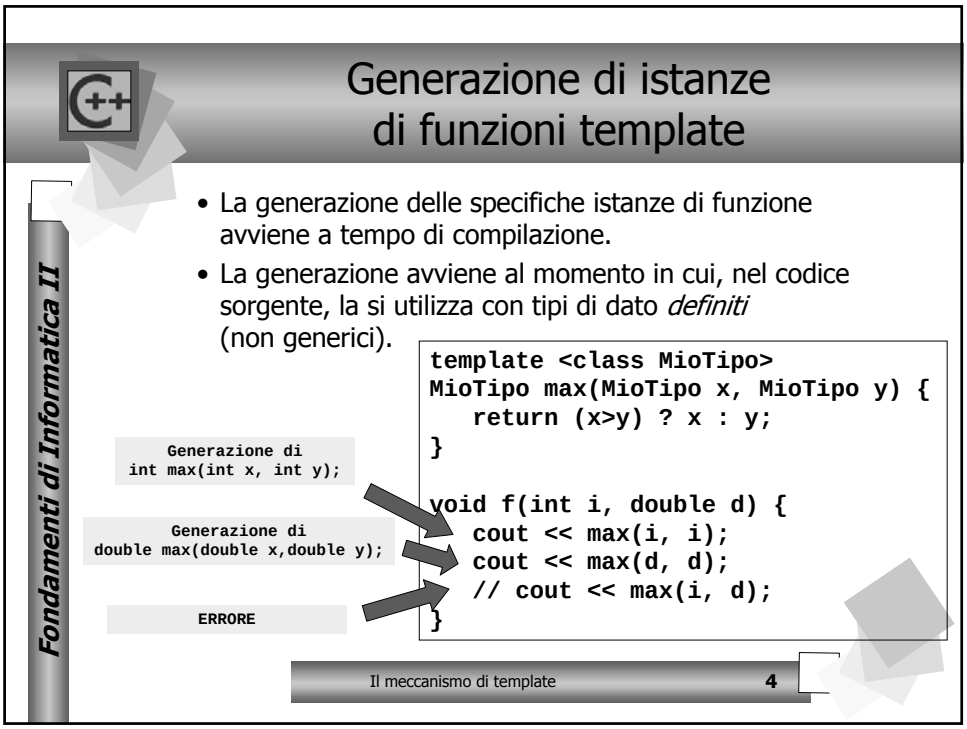
```
#define max(x,y)  ((x>y) ? x : y )
```

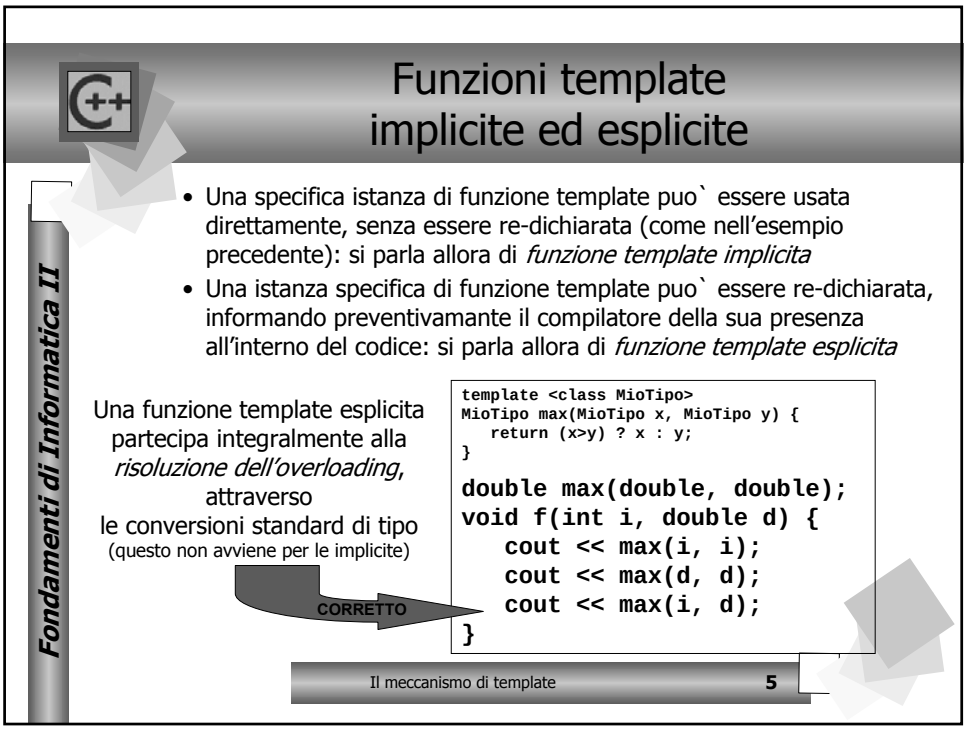
Problemi:
- si evita il controllo sui tipi
- si creano possibili errori sintattici

```
class foo {
  public:
    int max(int, int);
    // errore di sintassi
  [..]
}
```

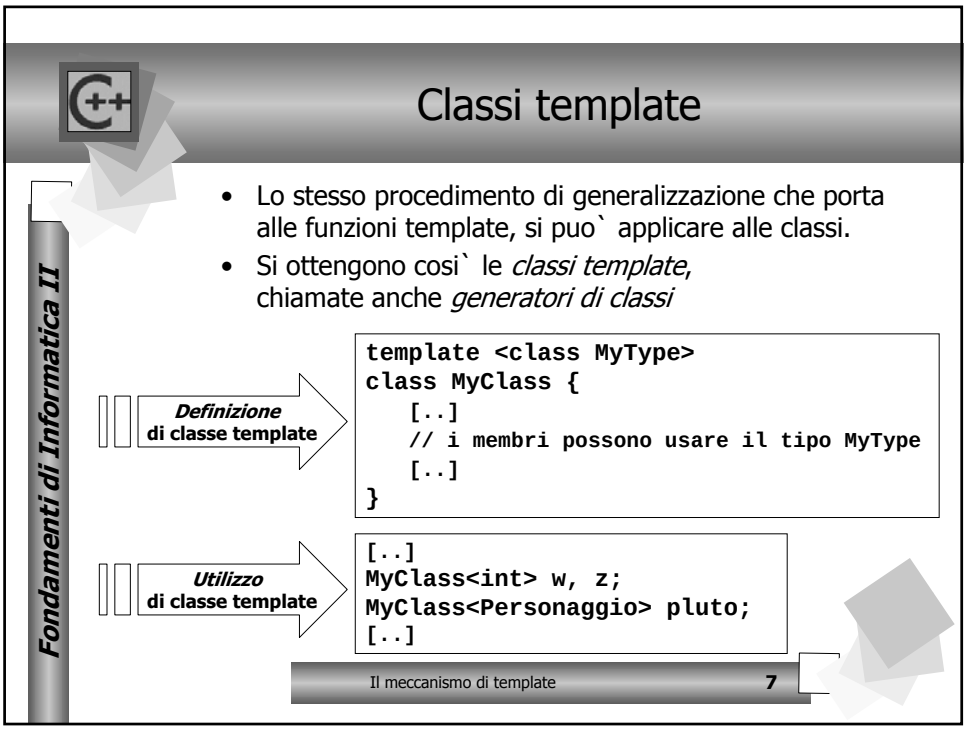
Perche' non usare semplici funzioni overloaded?


Funzioni template
Fondamenti di Informatica II
Il meccanismo di *template* permette di definire un *modello* per una famiglia di funzioni (overloaded) correlate, facendo in modo che lo stesso tipo di dato sia un parametro
Definizione di funzione template
template <class MioTipo>
MioTipo max(MioTipo x, MioTipo y) {
return (x>y) ? x : y;
}
> DEVE ESSERE DEFINITO PER MioTipo !!!
Utilizzo di funzione template
int i, j;
TipoBuono a, b;
[..]
j=max(i,0);
[..]
TipoBuono risult=max(a,b);
Il meccanismo di template 3


Generazione di istanze di funzioni template
Fondamenti di Informatica II
• La generazione delle specifiche istanze di funzione avviene a tempo di compilazione.
• La generazione avviene al momento in cui, nel codice sorgente, la si utilizza con tipi di dato *definiti* (non generici).
template <class MioTipo>
MioTipo max(MioTipo x, MioTipo y) {
return (x>y) ? x : y;
}
void f(int i, double d) {
cout << max(i, i);
cout << max(d, d);
// cout << max(i, d);
}
Generazione di int max(int x, int y);
Generazione di double max(double x,double y);
ERRORE
Il meccanismo di template 4


Funzioni template
implicite ed esplicite
Fondamenti di Informatica II
• Una specifica istanza di funzione template puo` essere usata direttamente, senza essere re-dichiarata (come nell'esempio precedente): si parla allora di *funzione template implicita*
• Una istanza specifica di funzione template puo` essere re-dichiarata, informando preventivamante il compilatore della sua presenza all'interno del codice: si parla allora di *funzione template esplicita*
Una funzione template esplicita partecipa integralmente alla *risoluzione dell'overloading*, attraverso le conversioni standard di tipo (questo non avviene per le implicite)
CORRETTO
template <class MioTipo>
MioTipo max(MioTipo x, MioTipo y) {
return (x>y) ? x : y;
}
double max(double, double);
void f(int i, double d) {
cout << max(i, i);
cout << max(d, d);
cout << max(i, d);
}
Il meccanismo di template
5


Funzioni template per tipi specifici
(specializzazione)
Fondamenti di Informatica II
• In alcuni casi speciali, la definizione della funzione template puo` essere non appropriata con particolari tipi.
• In questo caso, si puo` provvedere a definire una *specializzazione della funzione* per un tipo particolare.
• Il compilatore non genera istanze della funzione template per cui sono state definite specializzazioni.
funzione template generica
funzione template specializzata
#include <string.h>
template <class MioTipo>
MioTipo max(MioTipo x, MioTipo y) {
return (x>y) ? x : y;
}
char *max(char *x, char*y) {
return (strcmp(x, y)>0) ? x : y;
}
[..]
Il meccanismo di template
6


Classi template
Fondamenti di Informatica II
Lo stesso procedimento di generalizzazione che porta alle funzioni template, si puo` applicare alle classi.
Si ottengono cosi` le classi template, chiamate anche generatori di classi
Definizione di classe template
template <class MyType>
class MyClass {
[..]
// i membri possono usare il tipo MyType
[..]
}
Utilizzo di classe template
[..]
MyClass<int> w, z;
MyClass<Personaggio> pluto;
[..]
Il meccanismo di template 7


Istanze di classi template
Un'istanza di classe template è una classe specifica
Fondamenti di Informatica II
Il compilatore deve conoscere la definizione di una classe modello, per poterne generare specifiche istanze (non basta la dichiarazione): quindi occorre includere anche il file sorgente con la relativa definizione.
Come per le funzioni template, si possono definire specializzazioni di classi template.
Ogni classe specifica ha una sua copia di membri statici, distinti da quelli di altre classi istanza.
In modo analogo, le istanze di funzioni template hanno una loro copia di variabili automatiche static.
Il meccanismo di template 8

Operatore :: con classi template

Fondamenti di Informatica II

```
 /*** DICHIARAZIONE ***/

template <class TipoT>
class ClT {
  TipoT stato1;
  [..]
public:
  ClT(int);
  int   funz1(TipoT, int);
  TipoT funz2();
  [..]
}
```

```
        /*** DEFINIZIONI ***/

template <class TipoT>
ClT<TipoT>::ClT(int i) {..}

template <class TipoT>
int ClT<TipoT>::funz1(TipoT x, int i)
{..}

template <class TipoT>
TipoT ClT<TipoT>::funz2()
{..}
```

Quale sintassi per un metodo che restituisce un tipo definito internamente alla classe template?

Argomenti in classi template

Fondamenti di Informatica II

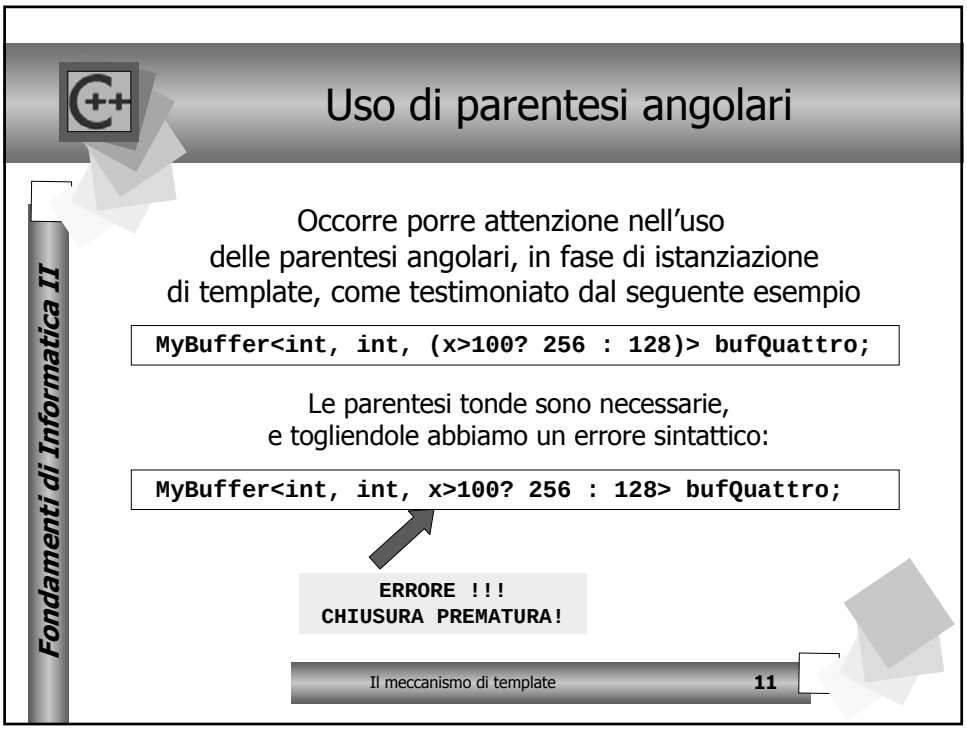
- La parola chiave "template" ammette argomenti multipli.

```
template <class T1, class T2, int dimensione = 128>
class MyBuffer {..};
```

- Il valore fornito come argomento *di tipo non template* deve essere un'espressione costante.

```
const int n = 1024;
int k = 256;
MyBuffer<int, double, 64+n/4> bufIntDoub; // corretto
MyBuffer<char, char, k*2> bufCharChar; // errore!
```

- Gli argomenti di una classe template, analogamente a quelli di una qualsiasi funzione, possono avere inizializzatori (valori default).


Uso di parentesi angolari
Fondamenti di Informatica II
Occorre porre attenzione nell'uso
delle parentesi angolari, in fase di istanziazione
di template, come testimoniato dal seguente esempio
MyBuffer<int, int, (x>100? 256 : 128)> bufQuattro;
Le parentesi tonde sono necessarie,
e togliendole abbiamo un errore sintattico:
MyBuffer<int, int, x>100? 256 : 128> bufQuattro;
ERRORE !!!
CHIUSURA PREMATURA!
Il meccanismo di template
11


Dichiarazioni friend
in classi template
Fondamenti di Informatica II
Vi possono essere vari usi
di dichiarazioni friend in classi template:
• Una classe o funzione normale: tutte le istanze di template hanno tale classe o funzione come friend.
• Un'altra classe (o funzione) template, con alcuni stessi argomenti: caso delicato.
• Un'altra classe (o funzione) template, con tutti argomenti diversi: tutte le istanze di template hanno tale classe o funzione come friend.
In ogni caso, è buona regola
evitare situazioni poco chiare
Il meccanismo di template
12