25 - Overloading e template di funzione (lez. 29-11-2024)

Nelle regole di risoluzione dell'overloading esistono delle *regole speciali* nel caso in cui troviamo dei template di funzione. N.B. il template non è una funzione, ma una sua specifica istanza sì, infatti quando cerco una funzione candidata per la risoluzione dell'overloading mi riferisco ad un'istanza del template.
Per decidere se una istanza / specializzazione di un template di funzione è un candidato valido dobbiamo verificare che sia possibile effettuare la deduzione dei parametri del template, durante questo processo si applicano solo le corrispondenze esatte. Se il processo di deduzione ha successo l'istanza del template diventa utilizzabile. Le regole specifiche per i template si applicano nella **terza fase**, quando occorre trovare la migliore funzione utilizzabile.

Ordinamento parziale dei template di funzione

I template di funzione con lo stesso nome e visibili nello stesso scope sono ordinati parzialmente rispetto a una "**relazione di specificità**". Definizione:

> **Definizione**
>
> Dato un template primario di funzione *X*, denotiamo con *istanze*(X) l'insieme di tutte le sue possibili istanziazioni.
> Si dice che il template di funzione X è più specifico del template di funzione Y se vale l'inclusione propria *istanze(X)* $\subset$ *istanze(Y)*.

Esempio:

```
//1: dice solamente che il primo argomento è di tipo T e il secondo di tipo U.
template <typename T, typename U> void foo(T t1, U u2);
//2: dice che il secondo argomento, oltre ad essere di tipo U, è un puntatore. E'
strettamente più specifico del primo.
template <typename T, typename U> void foo(T t1, U* u2);
//3: i parametri sono dello stesso tipo.
template <typename T> void foo(T t1, T t2);

//i template 2 e 3 non sono confrontabili
void foo(int, int*) //`e istanza di 2 ma non di 3
void foo(int, int) //`e istanza di 3 ma non di 2
```

Regole Aggiuntive

Quando si cerca tra le funzioni utilizzabili, la funzione migliore (se esiste), consideriamo:

- preferenza per i **template più specifici**
- preferenza per le **funzioni non templatiche**: se dopo aver eliminato le istanze meno specifiche si ottiene una situazione di ambiguità, allora si eliminano dalle istanze tutte le istanze di template.

```
//si considerino le seguenti dichiarazioni:
template <typename T, typename U> //#1
void foo( T a1, U a2);

template <typename T, typename U> //#2
void foo(T a1, U* a2);

template <typename T> //#3
void foo(T a1, T a2);

void foo(int* a1, int* a2); //#4

//chiamata
foo(42, 42);
//#2 e #4 non sono utilizzabili
//la funzione migliore è quella che si ottiene istanziando #3, preferito rispetto a
#1 per specificità
void foo<int>(int, int)

//chiamata
foo(&i, &i) //i di tipo int
//sono tutte utilizzabili e la #1 è scartata per specificità
//ci sarebbe ambiguità tra istanza #2
void foo<int*, int>(int*, int*)
//e istanza #3
void foo<int*>(int*, int*)
//e la funzione #4
//=> la migliore è la quattro (preferenza per non templatica)
```