12 - Conversioni implicite

La conversioni implicite di tipo avvengono automaticamente durante l'esecuzioni di un programma, queste conversioni consentono al compilatore di rendere compatibili tipi diversi di dati durante assegnazioni, passaggi di parametri o operazioni. Sono suddivise in 4 categorie principali:

1 - le corrispondenze "esatte":

sono quelle conversioni implicite che preservano il valore dell'argomento e si suddividono in tre sottocategorie:

1a. Identità (match perfetti):

Tecnicamente non è una conversione, ma viene considerata un caso speciale, si verifica quando il tipo del parametro e il tipo dell'argomento coincidono esattamente.
esempi: date le dichiarazioni `int i; const int& r = i`

argomento	tipo argomento	tipo parametro
5	int	int
i	int&	int&
&i	int*	int*
r	const int&	const int&
5.2	double	double

1b. Trasformazioni di lvalue:

Le trasformazioni di lvalue riguardano cambiamenti specifici che un lvalue subisce per adattarsi al tipo richiesto.
Tipi di trasformazioni:

1. **da lvalue a rvalue**: un lvalue viene utilizzato come un valore (es. passaggio per valore), si verifica quando siamo interessati al valore memorizzato in una locazione e non alla locazione in quanto tale;
2. **decay array-puntatore**: un array viene trattato come un puntatore al suo primo elemento;
3. **decay funzione-puntatore**: una funzione viene convertita in un puntatore alla funzione stessa;
   Esempi: date le dichiarazioni `int i; int a[10]; void foo();`

argomento	tipo argomento	tipo parametro	
i	int&	int	da lvalue a rvalue
a	int[10]	int*	decay array → puntatore
foo	void(int)	void(*)(int)	decay funzione → puntatore

1c. Conversioni di qualificazione:

Avvengono quando viene aggiunto il qualificatore `const` a un riferimento o a un puntatore. Questo accade per garantire la sicurezza del tipo, impedendo modifiche non intenzionali ai dati originali.
Esempi: data la dichiarazioni `int i`

argomento	tipo argomento	tipo parametro
i	int&	const int&
&i	int*	const int*

2 - Promozioni

Sono un tipo speciale di conversione implicita che preserva il valore dell'argomento. Sono progettate per adattare i tipi più piccoli o meno precisi a tipi più grani o più precisi a tipi più grandi o più precisi.

2a. Promozioni intere:

i tipi interi piccoli, come `char` e `short`, non sono direttamente rappresentabili nei registri del processore. Per questo motivo, vengono promossi a `int` o `unsigned int` prima di eseguire operazioni.
Esempi:

- `char` o `short` → `int` (o `unsigned int`)
- `bool` → `int`

2b. Promozioni floating-point:

I numeri in virgola mobile rappresentati come `float` vengono promossi a `double` per garantire maggiore precisione durante i calcoli.

2c. Promozioni delle costanti di enumerazione

Le costanti di enumerazione definite in C++2003 possono essere promosse al più piccolo tipo intero (almeno `int`) che possa contenerle.
Esempio:

```
enum Color { Red = 1, Green = 2, Blue = 3};
```

- `Red` viene promosso a `int`.

3 - Conversioni Standard

Includono tutte le altre conversioni implicite che non rientrano nelle categorie precedenti. Non garantiscono sempre la preservazione del valore e comprendono conversioni tra tipi numerici, tra puntatori e tra riferimenti.
Esempi

- da `int` a `long` (non è una promozione);
- da `char` a `double`;
- da `double` a `int`.

Conversioni tra puntatori e riferimenti

1. **Costante intera 0 e `nullptr`**
   - la costante 0 può essere convertita a un puntatore nullo (`T*`);
   - `nullptr` è il valore nullo di tipo `std::nullptr_t` , convertibile in qualsiasi tipo di puntatore.
2. **Puntatori a `void`** : un puntatore di tipo `T*` può essere convertito in `void*` , ma non viceversa.
3. **Up-cast**
   - da `D*` a `B*` (dove `D` è una classe derivata di `B`);
   - analogo per i riferimento `D&` → `B&` .

4 - Conversioni definite dall'utente

Comprendono due sottotipi:

4a. Uso implicito di costruttori:

Se una classe ha un costruttore non marcato come `explicit` , può essere usato implicitamente per convertire un tipo di dato in un altro.
Es.

```
struct Razionale {
    Razionale(int num, int den = 1); //conversione implicita da int a razionale
}
```

4b. Operatori di conversione:

Permettono di definire come un tipo utente può essere convertito in un altro tipo.
Es.

```
struct Razionale {
    operator double() const; //conversione da razionale a double
}
```

Promozioni

Type	Promoted To	Example
`short`	`int`	`short s = 5; int i = s;`
`char`	`int`	`char c = 'A'; int i = c;`
`unsigned char`	`int`	`unsigned char u = 255; int i = u;`
`float`	`double`	`float f = 3.14f; double d = f;`
`enum`	`int`	`enum Color { Red }; int i = Red;`

Conversioni

From Type	To Type	Method	Example
`int`	`float`	Implicit	`float f = 42;`

From Type	To Type	Method	Example
`char`	`int`	Implicit	`int i = 'A';`
`int`	`bool`	Implicit	`bool b = 42;`
`bool`	`int`	Implicit	`int i = true;`
Derived class ptr	Base class ptr	Implicit	`Base* b = derivedPtr;`