

# Tipi di Dato e Tipi di un Linguaggio

Sommario: 6 aprile, 2016

- Dati, Tipi di Dato e Tipi
- Tipi: Principi e Usi
- Sistemi di Tipi e Type Safety
- Equivalenza, Coercion and Cast
- Polimorfismo: Ad Hoc, Generico e di Sottotipo

# Dati, Tipi di Dato, Tipi: Dato

- **Dato:** La più semplice struttura **per introdurre valori** in un programma
- **Caratteristiche Distintive** sono i modi con cui possono essere usato nei programmi
  - Valori calcolabili: Espresi da espressione non atomica (i.e. variabile o literal);
  - Denotable values: Denotazioni in ambiente (in generale, dichiarazioni);
  - Storable values: (in genere) Valori di un modificabile;
  - Literal values: Espresi con literals.
- Ogni Linguaggio definisce quali modi sono usabili per i dati

## Example

```
int A[] = {2, 7, 12, 9};/ * corretto in C * /  
intSeq QSort(intSeq cc, int ord(int x, int y)){...};/ * errore in C * /  
A = {2, 7, 12, 9};/ * errore in C * /
```

- {2,7,12,9} esprime in C, un valore array solo nella dichiarazione
- una funzione in C è solo un valore denotabile: non può essere trasmessa, nè restituita, nè assegnata come valore memorizzabile


# Dati, Tipi di Dato, Tipi: Tipi di Dato

- **Tipi di Dato: Collezioni di valori omogenei** rispetto alle operazioni che possono essere loro applicate, nei programmi
  - **Caratteristiche Distintive** sono le operazioni applicabili
  - Una grande varietà. Si suddividono in:
    - **Scalari:** Sono valori atomici  
Esempi: *Interi, Virgola mobile, Complessi, Caratteri, Boolean, Enumerato, Intervallo, Void, Puntatore, ...*
    - **Strutturati:** Sono valori composti con operazioni per la selezione dei componenti  
Esempi: *Record, Array, Stringhe, Set, List, Collections, ...*
      - **Statici:** Sono allocati staticamente, alla dichiarazione/introduzione
      - **Dinamici:** Sono allocati dinamicamente mediante espressioni che ne estendono o riducono la struttura dei componenti
      - **Ricorsivi:** Sono tipi definiti in modo induttivo richiedono uno specifico meccanismo. Emulato in C mediante puntatori  
Esempi. `Type intList = Null | Cons int intList`
  - **Modificabili o no:** Il linguaggio stabilisce
    - Per gli scalari, quali siano, oppure no, assegnabili a variabili (ovvero quali siano memorizzabili)
    - Per gli strutturati, quali abbiano componenti modificabili.

# Dati, Tipi di Dato, Tipi: Tipi

- **Tipi: Strutture per classificare in modo univoco** il "ruolo" di ogni struttura usata in un programma con vari effetti e scopi:
  - **Evidenziare** l'organizzazione concettuale del programma  
Esempi. La presenza nel programma (a top-level<sup>1</sup>) di tipi per dati e operazioni che rispecchiano l'organizzazione dei valori e delle trasformazioni operate dal problema da meccanicizzare. Ad esempio: Il programma deriva introduce il tipo `parseTree`.
  - **Supportare il controllo** dell'allocazione di memoria dei dati  
Esempi. È il caso della funzione `malloc` in C e `new` in altri linguaggi, che usano i tipi per calcolare l'ammontare di memoria dinamica da allocare. Analogamente, il compilatore guarda ai tipi, nelle dichiarazioni, per la memoria statica.
  - **Verificare** proprietà statiche del programma
  - **Type Safety** ovvero Verifica contro stati di "stuck"

---

<sup>1</sup> ma per i principi della Programmazione Strutturata, ciò si ripete ad ogni livello con i sotto-problemi  4/9

# Dati, Tipi di Dato, Tipi: Tipi/2

- Tipi: **Strutture per classificare in modo univoco** il "ruolo" di ogni struttura usata in un programma con vari effetti e scopi:
  - **Evidenziare** l'organizzazione concettuale del programma
  - **Supportare il controllo** dell'allocazione di memoria dei dati
  - **Verificare** proprietà statiche del programma  
Esempi. È il caso della corrispondenza tra le entità definite e il loro uso nel programma, quali variabili assegnate e valore assegnabile, funzione dichiarata e sue invocazioni, struttura di un valore usato e operazioni applicate ad esso, ...
  - **Type Safety** ovvero Verifica contro stati di "stuck". Uno stato di "stuck" è uno stato che nessuna corretta esecuzione del programma dovrebbe raggiungere<sup>2</sup>.  
Esempi. È il caso del seguente programma C.

---

<sup>2</sup>Uno stato di "stuck" non è uno stato con anomalie quali eccezioni di programma o sistema di varia origine. Tali stati sono invece da considerarsi legali e raggiungibili anche da computazioni corrette.

# Type Safety

- **Type Safety** ovvero Verifica contro stati di "stuck". Uno stato di "stuck" è uno stato che nessuna corretta esecuzione del programma dovrebbe raggiungere<sup>3</sup>. Esempi. È il caso del seguente programma C.

## Example

```
int main (int argc, char * argv[]){ //cosa calcola?  
    int x = 10;  
    char a = ' e';  
    printf("Totale da pagare in euro : %2d\n", x + a);  
    return 0;  
}
```

*The program terminates computing :*  
Totale da pagare in euro : 111

- Dovrebbe essere evidente che il programmatore ha commesso un errore
- Nondimeno, questa è la peggiore situazione che può capitare nella programmazione
  - Il programma è errato ma appare corretto e termina in uno stato di "stuck" che sembra corretto
  - Come riconoscerlo? Come Impedirlo: Usare un sistema di tipi (sound)

---

<sup>3</sup> Uno stato di "stuck" non è uno stato con anomalie quali eccezioni di programma o di sistema. Stati con eccezione sono invece da considerarsi legali e raggiungibili anche da computazioni corrette.

# Sistema di Tipi

- Consiste di 3 strutture:
  - **Dominio dei Tipi** (che include i tipi basici del Linguaggio)  
Esempi.  $T = \text{Void} + \text{Int} + \text{Char} + \dots$
  - (Linguaggio delle) **Espressioni di Tipo** con cui possono essere definiti tipi (derivati)  
Esempi.  $T = \dots + T \times \dots \times T \rightarrow T + \dots$
  - **Regole (di inferenza)** con cui il sistema associa un tipo a ogni struttura del programma  
Esempi.  
$$\frac{E : \text{bool} \quad C : \text{Void}}{\text{while } E \text{ do } C : \text{Void}} \quad \frac{E : \text{bool} \quad E_{\text{then}} : T1 \quad E_{\text{else}} : T2 \quad T1 = T = T2}{E ? E_{\text{then}} E_{\text{else}} : T}$$
- Se il linguaggio ha Sistema di Tipi *sound*, allora  
Un programma è *safe* da "stuck" se il Sistema assegna a ogni struttura 1 e 1 solo tipo.

# Relazioni e Proprietà sulla struttura dei Tipi

Intervengono quando non c'è identità tra tipo atteso e tipo effettivo.

Esempio.  $\{T1 \ x; \dots; x = E\}$ . Il tipo atteso per  $E$  è  $T1$ , ma  $E$  ha tipo  $T2 \neq T1$ .

- **Equivalence.** Sia  $x:T$  and  $y:T'$ . Quando  $x$  e  $y$  hanno stesso tipo?
  - **Nominal.** Ogni tipo è uguale solo a se stesso.  
Esempi. `{type T1 = int; T1 x; int y; x=y}` contiene un errore di tipo
  - **Structural.** Stessa espressione di Tipo quando i nomi sono rimpiazzati dalle definizioni.  
Esempi. `{type T1 = int; T1 x; int y; x=y}` sequenza corretta
- **Compatibilità e Sottotipo**
  - $T$  è compatibile con  $S$  se (un valore di tipo)  $T$  può essere usato quando è atteso  $S$ .  
Esempi. Sottotipi  $T \subset S$  in Java
- **Coercion:**  $T$  è compatibile con  $S$  ma i valori di  $T$  devono cambiare rappresentazione per essere usati come  $S$ ; Esempi. `int` e `float` in `C`
- **Cast:** compatibilità senza necessità di cambiamenti sulla rappresentazione;  
Esempi. oggetto di una sottoclasse e una sua superclasse in Java



# Relazioni e Proprietà sulla struttura dei Tipi /2

Intervengono quando non c'è identità tra tipo atteso e tipo effettivo.

Esempio.  $\{T1 \ x; \dots; x = E\}$ . Il tipo atteso per E è T1, ma E ha tipo  $T2 \neq T1$ .

- **Overloading** (Polimorfismo ad Hoc) Differenti tipi e valori funzionali per uno stesso (Function) identifier;  
Esempi.  $+$ ,  $*$ ,  $\dots$  per int e float hanno definizioni completamente diverse ma si chiamano nello stesso modo.
- **Parametric Polymorphism** (Polimorfismo generico) Le espressioni di tipo includono variabili quantificate universalmente su i tipi del linguaggio.  
**Permettono di esprimere insiemi (anche infiniti) di tipi.** Nei sistemi di tipi, un ordinamento parziale permette sempre di individuare l'insieme di tipi più generale tra quelli associabili ad ogni struttura del programma.  
**Esempi.**  $Seq(t)$ ,  $\langle t \rangle Seq(t)$   $QSort(Seq(t)S, t \times t \rightarrow bool L)$  nel QuickSort;  
Una sola  $\langle t \rangle Swap(\dots t x, \dots t y)$  su un generico  $t$  invece che tante Swap per diversi tipi utilizzati nel programma.  
**Meccanismo fondamentale per riuso di codice** (presente in tutti i linguaggi di ultima generazione)
- **Polimorfismo di sottotipo.** Polimorfismo in combinazione con i sottotipi.  
Esempi. Java prime versioni solo Polimorfismo di sottotipo, successive versioni Polimorfismo generico e di sottotipo (lo useremo)