



Programmazione 1: Tipi di Dato

Michele Nappi, Ph.D

mnappi@unisa.it



Tipi di Dato

- Potere Espressivo di un LP
 - Tipi di dato che nel Linguaggio si possono:
 - Esprimere e Manipolare
 - Operazioni esprimibili direttamente o per composizione
 - Set Istruzioni di LP
 - Funzioni definibili dall'utente



Tipi di Dato (cont.)

- Definizione
 - Un tipo di dato T astratto è definito da:
 - Un dominio D di valori
 - Un insieme (finito) F di operatori $op_1 op_2 \dots op_k$
 - Manipolazione dei valori di D
 - » $op_i : D \times D \rightarrow D$ (se è un operatore binario)
 - » $op_i : D \rightarrow D$ (se è un operatore unario)
 - Esempio
 - Tipo int
 - » $D \equiv \mathbb{Z}$ dove \mathbb{Z} è l'insieme degli interi nell'intervallo $[-\infty, +\infty]$
 - » $F \equiv \{+, *, /, >, <, ==\}$
 - » Operatori aritmetici e logici (predicati)



Tipi di Dato (cont.)

- Tipologia Tipi di Dato (TD)
- TD Primitivi
 - Basati direttamente sulla rappresentazione HW (int, char, real, boolean)
 - Supportano i TD del LP
- TD LP
 - Supportati dal LP
 - Tipi strutturati (Array)
- Livello Utente
 - Definiti dall'utente usando le specifiche del LP
 - Record, Grafi, Liste (alcuni potrebbero essere supportati da LP)



Tipi di Dato (cont.)

- Classificazione
 - Elementari
 - Dominio di elementi atomici ed omogenei
 - Interi, caratteri, reali booleani
 - Coincidono con TD Primitivi
 - Strutturati
 - Combinazione logica di TD elementari e strutturati
 - Dominio di elementi non atomici ed eterogenei
 - Array, Record, Grafi, Liste (supportati da LP o definiti dall' utente)



Tipi di Dato (cont.)

- Astrazione dati
 - Indipendenza dalla rappresentazione (incapsulamento/implementation hiding)
 - La *specifica astratta* definisce il comportamento di un oggetto (TD nel nostro caso) indipendentemente dalla sua implementazione
 - La *rappresentazione concreta* definisce l'implementazione dell'oggetto
 - Esempio
 - *Specifiche astratte*: Sequenza dei numeri primi nell'intervallo [7,19]
 - *Rappresentazione Concreta*: $\text{int primo}[5]=\{7,11,13,17,19\}$



Tipi di Dato (cont.)

- Astrazione dati
 - Indipendenza dalla rappresentazione.
 - *Un programma dovrebbe essere progettato in modo tale che sia possibile modificare la rappresentazione di un oggetto senza modificare il resto del programma*
 - Se l'oggetto è un TD
 - Astrazione Dati
 - Se l'oggetto è una funzione (procedura)
 - Astrazione Funzionale



Tipi di Dato (cont.)

- Dichiarazione del TD
 - Si crea un'associazione permanente tra il TD, l'identificatore e l'indirizzo assegnato durante l'esecuzione del codice
 - Ciò che cambia (eventualmente) è il contenuto della locazione in sintonia con il concetto di variabile



Variabili, TD e Memoria

- La memoria è un insieme di locazioni

100	
101	
102	3
103	
104	
105	
106	
107	
108	
109	

i

```
int i; /* dichiarazione */  
  
i = 3; /* inizializzazione */
```

- Ogni variabile ha un *nome*, un *tipo*, un *valore* ed una *locazione*.
 - I nomi di variabili corrispondono alle locazioni nella memoria del computer.
- Nell'esempio:
 - Il nome della variabile è **i**
 - La variabile **i** è di tipo **int**
 - La variabile **i** ha valore **3**



Variabili, TD e Memoria

(cont.)

```
/* lettura e scrittura di variabili */
#include <stdio.h>
int main(void)
{
    int i, j, sum;
    i = 3;
    printf("Inserisci un intero:\n");
    scanf("%d", &j);
    sum = i + j;
    printf("i = %d\nj = %d\tsomma = %d\n", i, j, sum);
    j = i;
    sum = i + j;
    printf("i = %d\nj = %d\tsomma = %d\n", i, j, sum);
    return 0;
}
```

Inserisci un intero: -5

i=3 j=-5 somma=-2
i=3 j=3 somma=6

102		
103	-5	j
104		
105		
106	-2	sum
107		
108	3	i
109		

102		
103	3	j
104		
105		
106	6	sum
107		
108	3	i
109		



Tipi di Dato (cont.)

- **Overloading** (sovraffidato)
 - Ad uno stesso operatore è assegnata nell'ambito di LP una semantica differente a seconda del tipo a cui è applicato
 - Il C consente l'overloading
 - Esempio: l'operatore somma (+)
 - Se applicato ad un **int**: $+ : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$
 - Se applicato ad un **float**: $+ : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$



Tipi di Dato (cont.)

Compatibilità

- Siano T_1 e T_2 due TD:
 - T_1 ha operatori $op_i : D_1 \times D_1 \rightarrow D_1$
 - T_2 ha operatori $op_i : D_2 \times D_2 \rightarrow D_2$
 - T_1 è compatibile con T_2 se $D_1 \subseteq D_2$
 - Se op_i è definito per il tipo T_2 ossia $op_i : D_2 \times D_2 \rightarrow D_2$ allora le seguenti notazioni sono corrette:
 - $op_i : D_1 \times D_2 \rightarrow D_2$
 - $op_i : D_2 \times D_1 \rightarrow D_2$



Tipi di Dato (cont.)

- Compatibilità
 - *Coercion* (Forzatura)
 - Le variabili di tipo T_1 sono considerate automaticamente di tipo T_2
 - Se entrambi gli argomenti sono di tipo T_1 :
 - L'operatore è definito solo per T_2
 - $op_i : D_1 \times D_1 \rightarrow D_2$
 - L'operatore è definito sia per T_1 che per T_2 (overloading)
 - $op_i : D_1 \times D_1 \rightarrow D_1$



Type Checking

- Controllo dei Tipi
 - Garantisce che le operazioni di un programma siano applicate in modo proprio
 - Previene gli errori
 - Esempio: un float gestito come un int determina un errore di tipo
 1. int pippo, i, j;
 2. float pluto= 0.3456;
 3. pippo=pluto; **type checking error (pippo=0)!!!**
 4. i=j/pippo;
 - Un programma si dice *safe* se è sicuro rispetto ai tipi
 - Esecuzione senza errori di tipo



Type Checking (cont.)

- Controllo dei Tipi
 - Compilatore o Interprete
- Statico
 - Il controllo avviene durante la traduzione del codice sorgente
- Dinamico
 - Il controllo viene effettuato durante l'esecuzione
 - » Si inserisce nel programma eseguibile del codice preposto al controllo durante l'esecuzione



Type Checking (cont.)

- Controllo dei Tipi
 - Statico
 - Compilazione rallentata dal controllo dei tipi
 - Debugging lento
 - Dinamico
 - Il codice aggiuntivo consuma spazio e tempo
 - Esecuzione meno efficiente
 - Un eventuale problema di tipo si manifesta solo durante l'esecuzione
 - Poiché parti di codice possono essere usate solo di rado se il problema è presente in tali sezioni.....



Type Checking (cont.)

- L'efficacia del controllo statico e l'onerosità del controllo dinamico determinano che le implementazioni dei LP controllino solo quelle istanze verificabili staticamente
 - Le proprietà dipendenti da valori calcolati a tempo d'esecuzione vengono tralasciate
 - Divisioni per zero
 - Conformità di un indice rispetto alle reali dimensioni di un array



Type Checking (cont.)

- Type Checking System
 - Safe
 - Programma eseguito senza errori di tipo
 - Forte
 - Accetta solo espressioni sicure: le espressione accettate saranno valutate senza errori di tipo. Accetta un sottoinsieme dei programmi *safe*
 - Debole
 - Non Forte