Tipi, dichiarazioni e definizioni di variabili

Definizione di tipo di una variabile

Il **tipo** è un termine di classificazione che raggruppa tutte quelle variabili che sono **memorizzate nello stesso modo** e a cui si applica lo **stesso insieme di operazioni**.

Tipi di dato

»Consente di esprimere la natura del dato
»Indica il modo con cui verrà interpretata la sequenza di bit che rappresenta il dato
»La stessa sequenza può rappresentare un intero o un carattere
»Determina il campo dei valori che un dato può assumere
»Specifica le operazioni possibili sui dati
»Ogni tipo di dato ha
- – Un nome
  - int, double, char, ….
- – Un insieme di valori letterali possibili
  - 3, 3.1, 'c', ….
- – Un insieme di operazioni lecite
  - +, *, /, %, ……

»Java ha
- – Tipi Primitivi (*predefiniti*)
- – Tipi Reference (costituiscono *puntatori* a degli oggetti)

Definizione obbligatoria degli identificatori

Un'istruzione di **dichiarazione** si limita ad informare il compilatore che un certo **identificatore** appartiene a un certo **tipo**, ma può non essere considerata in fase di esecuzione del programma.

Quando una **dichiarazione** comporta anche un'operazione eseguibile, allora si dice che è anche una **definizione**.

Ad esempio l'istruzione: **int error_number;**

è anche una **definizione**, in quanto non si limita ad informare il compilatore che la variabile **error_number** è di **tipo int**, ma crea la variabile stessa, allocando un'apposita area di memoria.

Per meglio comprendere la differenza fra **dichiarazione** e **definizione**, si considerino le seguenti regole:

- tutte le **definizioni** sono anche **dichiarazioni** (ma non è vero il contrario);
- deve esserci una ed una sola **definizione** per ogni **identificatore** che appare nel programma (o meglio, per ogni **identificatore** che appare in uno stesso **ambito**, altrimenti si tratta di **identificatori** diversi, pur avendo lo stesso **nome**), mentre possono esserci più **dichiarazioni** (purché non in contraddizione fra loro);
- un **identificatore** deve essere **dichiarato** prima del suo utilizzo, ma può essere **definito** dopo (o altrove, come vedremo *nel caso di metodi astratti cioè senza corpo*);
- la semplice **dichiarazione** di una variabile di **tipo *primitivo*** è sempre anche una **definizione**, in quanto comporta l'allocazione di un'area di memoria:

```
// dichiarazione
 int x;              // dichiara nome e tipo e contemporaneamente riserva spazio in RAM
                     // il cui indirizzo sarà referenziato con quel nome
// inizializzazione
 x = 10;             // operazione di assegnamento che imposta il valore

// dichiarazione ed inizializzazione
 int y = 15;
```

*Con la **dichiarazione**, l'**entità** viene associata all'identificatore (nome della variabile o della funzione) e ne viene dichiarata la tipologia, ma non viene ne' riservata la memoria per le variabili, ne' specificato il codice (cioè il corpo) per le funzioni.*

*Con la **definizione**, oltre ad associare l'identificatore alla entità e definirne la tipologia (es: variabile di tipo int), viene riservata memoria per le variabili, o specificato il codice per le funzioni.*

Le definizioni specificano il codice o i dati descritti dal nome. Il compilatore necessita della definizione per allocare spazio di memorizzazione per l'elemento che viene dichiarato.

Definizione con Inizializzazione

Un **inizializzatore** è un'espressione che definisce il **valore iniziale** assunto dalla variabile, ed è separato dal resto della **definizione** dall'**operatore** "**=**".

Quindi, ricapitolando (nel caso che l'**identificatore** sia il nome di una variabile):

- la semplice **dichiarazione** assegna un **tipo** alla variabile;
- la **definizione** crea la variabile in memoria, ma non il suo contenuto, che rimane, per il momento, indefinito (forse resta quello che c'era prima nella stessa locazione fisica di memoria);
- la **definizione** con **inizializzazione** attribuisce un valore iniziale alla variabile definita.

  Es. **double peso = 57.65;**

n.b. un'**inizializzazione** è concettualmente diversa da un'**assegnazione** pur usandolo stesso operatore.

In informatica l'**operatore di assegnamento** imposta (***inizializza***) o **reimposta** il valore memorizzato nella posizione di memoria associata a una variabile

inizializzazione**: assegnazione di un valore ad una variabile precedentemente **definita
***assegnazione**: istruzione che permette di valorizzare una variabile (quindi, inizializzarla) o di cambiarne il valore attuale.*

» Prende il valore che si trova alla destra (rvalue) e lo copia nell'entità di sinistra (l-value)
» l-value deve essere il nome di una variabile

```
x = 4      //espressione OK
4 = x      //espressione NOT OK
```

» Nel caso di **tipi primitivi** della variabile viene **copiato il valore**
» Nel caso di **tipi reference** viene copiato il **riferimento dell'oggetto** *(cioè l'indirizzo alla area di RAM)*

I valori di **inizializzazione** possono essere dati non solo da costanti, ma anche da **espressioni che includono variabili definite precedentemente**.

Es. **int lordo = 45; int tara = 23; int netto = lordo-tara;**

Quindi, ricapitolando, nel caso di definizione di una variabile, il compilatore :

*- noto di che **tipo** è la variabile* *(**dichiarata** di un certo tipo e **nome**)*

*- dal tipo deduce le **dimensioni in bytes***

*- **riserva spazio in memoria** associato ad un **indirizzo*** *(associa il **nome** all'indirizzo)*

*ora la variabile ha **valore** INDEFINITO (forse quello già in memoria RAM)*