Variabili e tipi di dato in C

Alessandra Giordani
agiordani@disi.unitn.it
Lunedì 29 marzo 2010
http://disi.unitn.it/~agiordani/

Variabili e valori

- Una variabile è un nome logico a cui e assegnato un valore.
- Quest'ultimo può cambiare nel corso dell'esecuzione di un programma.

```
…
x = 3
…
x = 7
```

Ciclo di vita di una variabile

- **Dichiarazione**: si informa il compilatore che esiste una certa variabile;
- **Inizializzazione/assegnamento**: si imposta il valore attuale della variabile (il termine “inizializzazione” si riferisce al primo assegnamento di una variabile);
- **Utilizzo**: si utilizza il valore associato per calcolare espressioni arbitrariamente complesse.

Il linguaggio C è

- *esplicitamente tipato*: occorre esplicitamente associare ad ogni variabile il tipo di dato che essa rappresenta
- *staticamente tipato*: il tipo di una variabile non puo cambiare durante il suo ciclo di vita;
- *fortemente/debolmente tipato*: non è possibile mischiare tipo di dato diversi nella stessa espressione, o se è possibile è richiesto che le conversioni siano esplicite (casting, lo vedremo in seguito). Però, in alcuni casi questa conversione esplicita non è necessaria.

Dichiarazione di una variabile

- Prima di poter essere utilizzata in qualsiasi altro modo, una variabile deve essere dichiarata associando al suo nome un tipo di dato.
- La più semplice istruzione di dichiarazione ha la forma:

```
<tipo di dato> <nome della variabile> ;
```

- dove il ; indica la fine dell'istruzione (tutte le istruzioni in C terminano con ;).

Nome delle variabili

- Il `<nome della variabile>` è un nome di comodo che scegliamo di associare ad una variabile.
- Il nome può avere qualsiasi lunghezza > 1
- Teniamo presente che il C è case sensitive, quindi i nomi `aBC` e `Abc` sono due nomi diversi!
- Un nome può contenere solo:
  - lettere minuscole (a-z);
  - lettere maiuscole (A-Z);
  - cifre (0-9);
  - il carattere underscore (_).
  - Tutte le combinazioni di questi caratteri sono valide, tranne quelle in cui il primo carattere e una cifra

Nomi di variabili validi e non

- Nomi di variabili validi:
  - aBC
  - a
  - X
  - x2
  - ciao_sono_io
  - _234
- Mentre i seguenti non sono nomi validi:
  - 2aBC
  - 1
  - Ciao*()@

Nomi di variabili non validi

- Inoltre, non sono ammissibili come nomi di variabili tutti quei nomi che appartengono ad una lista di nomi riservati, già utilizzati dal C come costrutti sintattici o operatori:

auto	char	do	entry	for	const
if	register	sizeof	switch	unsigned	signed
break	continue	double	extern	goto	volatile
int	return	static	typedef	while	
case	default	else	float	enum	
long	short	struct	union	void	

Scelta del nome di una variabile

- Il nome che decidiamo di assegnare ad una variabile è arbitrario, ma è un principio di buona programmazione che il nome rietta l'uso che intediamo fare della variabile.
- Ad esempio, **`delta`** è un buon nome per rappresentare il delta della formula risolutiva di un'equazione di secondo grado,
- mentre **`num_ruote`** è piu adatto per rappresentare il numero di ruote di un veicolo.

Tipi di dato fondamentali

- int e il tipo di dato che consente di rappresentare numeri interi all'interno di un programma.
  - Su un architettura a 32 bit, un int è rappresentato in complemento a 2 utilizzando 4 byte;
- float permette di rappresentare numeri in virgola mobile
  - usando 4 byte con una precisione di circa 7 cifre decimali;
- double permette di rappresentare numeri decimali in virgola mobile in doppia precisione
  - usando 8 byte con una precisione di circa 15 cifre decimali;
- char permette di rappresentare caratteri alfabetici.
  - usando un byte. La sequenza di bit che rappresenta un carattere puo anche essere interpretata come un intero senza segno

Esempi di dichiarazioni

```
char a;        // dichiara una variabile di tipo “carattere” chiamata “a”
int b;         // dichiara una variabile di tipo “intero” chiamata “b”
int ciccio;    // dichiara una variabile “intera” chiamata “ciccio”
float decim; // dichiara una variabile “decimale” chiamata “decim”
double x;      // dichiara una variabile “decimale a doppia precisione”
               chiamata “x”
```

Non è possibile dichiarare più di una variabile con lo stesso nome!

Assegnamento di una variabile

- Dopo aver dichiarato una variabile è possibile assegnare un valore alla variabile.
- L'assegnamento di un valore ad una variabile ha la forma:

  `<nome variabile> = <espressione>`
- dove = è l'operatore di assegnamento del C (equivalente a ← in pseudocodice) e

  `<espressione>` può essere…

Assegnamento con espressione

- un valore costante: in questo caso, si copia il valore all'interno della variabile;
- un nome di variabile: in questo caso, si copia nella variabile il cui nome è a sinistra dell'= il valore memorizzato nella variabile a destra;
- un'espressione il cui risultato sia compatibile con il tipo di dato della variabile il cui valore vogliamo modicare. L'espressione viene valutata prima di effettuare l'assegnamento. In questo modo, e possibile scrivere a destra dell'uguale espressioni che contengano lo stesso nome che compare a sinistra.

Esempi di assegnazione

```
int a, b;  // Si possono dichiarare piú variabili dello stesso tipo in una sola volta!
char c, d;
double e, f;
a = 7;
b = 10.7;  // Se assegniamo una costante decimale ad una variabile intera, la parte
           // decimale viene automaticamente troncata, cioé in questo caso nella
           // variabile immagazziniamo il valore 10!!
c = 'C';
d = ';';   // anche ';' é un carattere
e = 14567; // Se assegniamo un intero ad una variabile decimale, il numero viene
           // automaticamente convertito in intero
f = 456.789;
```

- In C il separatore delle cifre decimali è il punto (.) e non la virgola (,)
- Gli apici (‘) sono necessari per far capire al compilatore che intendiamo il carattere ‘a’ piuttosto che il contenuto della variabile a
- La possibilita di assegnare valori decimali ad una variabile dichiarata intera (e viceversa) e un'altra delle ragioni per cui il C può essere considerato un linguaggio debolmente tipato.

Esempi di assegnazione (cont)

```
int a, b;  // Si possono dichiarare piú variabili dello stesso tipo in una sola volta!
char c, d;
double e, f;
a = 7;
b = 10.7;  // Se assegniamo una costante decimale ad una variabile intera, la parte
           // decimale viene automaticamente troncata, cioé in questo caso nella
           // variabile immagazziniamo il valore 10!!
c = 'C';
d = ';';   // anche ';' é un carattere
e = 14567; // Se assegniamo un intero ad una variabile decimale, il numero viene
           // automaticamente convertito in intero
f = 456.789;
```

```
int somma;
float media;
somma=a;
somma=somma+b;
media=somma/2;
```

Dichiarazione e inizializzazione

- È possibile combinare le 2 operazioni in una sola istruzione:

```
int a = 50, b, c; /* Dichiara tre variabili, assegna un valore solo alla prima */
char d, e = 'X', f = 'Z'; /*Dichiara 3 variabili, assegna un valore alle ultime due */
d = f; /* Assegna a d il valore attuale di f, cioé 'Z' */
```

- Cosa succede se non inizializziamo una variabile nè le assegnamo un valore prima di accdervi? Che valori hanno b e c?

Stringhe

- Un altro tipo di insieme che vorremmo poter rappresentare è quello delle stringhe di caratteri, cioè sequenze di caratteri, per poter comporre parole e frasi (ad esempio, per visualizzare messaggi sullo schermo).
- In C la rappresentazione delle stringhe è un argomento complesso. Per il momento limitiamoci ad considerare una stringa costante in C come una sequenza di caratteri compresi tra doppi apici ("). Ad esempio:
  - “ciao mondo!"
  - "questa è una stringa"
  - "per favore, inserisci un numero"

Stringhe (cont)

- Alcuni caratteri non sono rappresentabili con un singolo carattere (ad esempio, il carattere di fine riga che ci fa visualizzare un “a capo”)
- Delle sequenze speciali (chiamate escape sequences, sequenze di escape) ci consentono di inserire questi caratteri.
- Ad esempio, il carattere di fina riga puo essere rappresentato con '\n‘
- Quindi, la stringa "questa e una stringa\n" termina con un carattere di fine riga.

printf() – stringa costante

- Introduciamo le stringhe perchè ci servono per utilizzare la funzione printf che ci consente di stampare messaggi a video.

La funzione printf (che studieremo e deniremo più avanti) puo essere invocata in due modalità:

- printf(<una stringa costante>) stampa a video una stringa.
  - Ad esempio: `printf("Hello World!");`
  - visualizza sullo schermo il messaggio \Hello World!".

printf() - template

- Ma printf può anche essere usata per stampare il contenuto di una variabile
- all'interno della stringa possiamo inserire delle sequenze di caratteri speciali a cui, in fase di stampa, l'interprete sostituirà il valore attuale di una variabile.
- In questo caso, dobbiamo anche dire alla funzione di quale variabile vogliamo stampare il valore:
  - per stampare una variabile int, la sequenza speciale è %d
  - per stampare una variabile float, la sequenza speciale è %f
  - per stampare una variabile char, la sequenza speciale è %c

printf() - template

- Queste sequenze possono apparire ovunque nella stringa (template) che passiamo a printf.
- Il valore della variabile verra stampato a video nella stessa posizione.
- Per dire alla funzione printf di quali variabili vogliamo stampare il valore, possiamo aggiungere alla funzione tanti argomenti quante sono le variabili di cui intendiamo stampare il valore.

printf() - Esempio

```
char a = 'B';
int x = 1000, y = 20000;
float k = 50.5;
printf("la variabile a vale: %c", a);
printf("la variabile x vale: %d", x);
printf("la variabile y vale: %d", y);
printf("la variabile k vale: %f", k);
printf("le variabili: k,x,a valgono rispettivamente:
                %f,%d,%c", k,x,a );
```

- copiare nel main vuoto e salvare come file .c, compilare ed eseguire per vedere cosa accade
- NB. Il carattere " viene copia/incollato sbagliato in gedit, riscriverli a mano

Array

- Gli array sono sequenze di dimensione finita di variabili omogenee
- Ciascun elemento di un array è una variabile
- Tutti gli elementi di un array sono rappresentati contiguamente in memoria
- La sua dimensione (numero di elementi) e definita in fase di dichiarazione
- La sua dimensione non cambia durante il suo ciclo di vita

Array e Matrici

- E' possibile definire array uni-dimensionali (vettori) e bi-dimensionali (matrici)
- Ciascun elemento puo essere acceduto mediante il suo indice
  - vettore[indice]
  - matrice[indice][indice]
- Il primo elemento di un array di dimensione n ha indice 0, l'ultimo n-1
- NB: Le funzioni non possono ritornare array!

Array e matrici - dichiarazione

```
int     a[5];   // dichiara un array di 5 variabili intere

char    b[2];   // dichiara un array di 2 caratteri

double  c[] = {10, 5, 0.5} // dichiara un array di double di
                           // lunghezza 3 e ne inizializza
                           // i valori

int matrice1[4][2];   // dichiara una matrice di interi con
                      // 4 righe e due colonne
```

NB: L'inizializzazione può avvenire in fase di dichiarazione

Accesso agli elementi dell'array

- L'accesso in lettura/scrittura agli elementi di un array avviene per mezzo degli indici

```
int data[3];
data[0]=29;
data[1]=03;
data[2]=10;
printf("Oggi è il giorno %d-%d-%d\n",
       data[0], data[1], data[2]);
```

Accesso agli elementi dell'array

- Per un array di dimensione n gli indici vanno da 0 a n-1
- Cosa accade se usiamo un indice non valido? Provate a compilare ed eseguire

```
int data[3];
printf("Contenuto all'indice 3: %d\n", data[3]);
```

Predicati

- Una predicato è un costrutto linguistico del quale si può asserire o negare la veridicità. Il suo valore di verità dipende dall'istanziazione di alcune variabili
  - "*l'indice* è minore della *dimensione del vettore*"
- La valutazione di un predicato è l'operazione che permette di determinare se il predicato è vero o falso, sostituendo alle variabili i loro valori attuali
  - "*l'indice* è minore della *dimensione del vettore*"
  - "*3* è minore di *3*" → FALSO

Predicati semplici e composti

- I predicati che contengono un solo operatore relazionale sono detti semplici
- Dato un predicato p, il predicato not p, detto opposto o negazione logica di p, ha i valori di verità opposti rispetto a p
- Dati due predicati p e q, la congiunzione logica p and q è un predicato vero solo quando p e q sono entrambi veri, e falso in tutti gli altri casi
- Dati due predicati p e q, la disgiunzione logica p or q è un predicato falso solo quando p e q sono entrambi falsi, e vero in tutti gli altri casi
- I predicati nei quali compare almeno un operatore logico, not, and, or, sono detti composti

Il costrutto IF

- Usato per rappresentare il branching dei diagrammi di flusso

```
if (<condizione>) { /* blocco_if ... */ }
[ else { /* blocco_else ... */ } ]
```

- Esegui blocco if solo se <condizione> != 0
- Se presente, esegui blocco else solo se <condizione> == 0
- I predicati sono condizioni
- È possibile creare condizioni composte

Esempio

```
int data[]={29,3,10};
int i,n;
n=3; //dimensione vettore
i=1; //indice vettore, punta al 2° elemento
if (data[i]==3)
  printf("Siamo in marzo");
if (i>=0)
  printf(", giorno %d\n",data[i-1]);
i=i+2;
if ((i>0)&&(i<n))
  printf("Contenuto all'indice %d: %d", i,data[i]);
else printf("Indice non valido\n");
```