Variabili intere, funzioni e input/ output da tastiera

Alessandra Giordani

agiordani@disi.unitn.it

Lunedì 19 marzo 2012

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.

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 è 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 modificare. L'espressione viene valutata prima di effettuare l'assegnamento. In questo modo, è possibile scrivere a destra dell'uguale espressioni che contengano lo stesso nome che compare a sinistra.

Esempi di assegnazione

```
int secondi;

int ore;

ore = 3;   // assegna alla variabile ore
           // il valore 3

secondi = ore * 60;
           // assegna alla variabile secondi
           // il valore contenuto nella
           // variabile ore moltiplicato x60
```

Dichiarazione e inizializzazione

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

```
int ore = 3;
int secondi = ore * 60;
```

- Cosa succede se non inizializziamo una variabile nè le assegnamo un valore prima di accedervi? Che valori hanno?

```
int ore;
int secondi = ore * 60;
```

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 definiremo 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.

Assegnazione - esempio.c

- Inizializza le tre variabili e stampane il contenuto
- Verifica il risultato dell'assegnazione

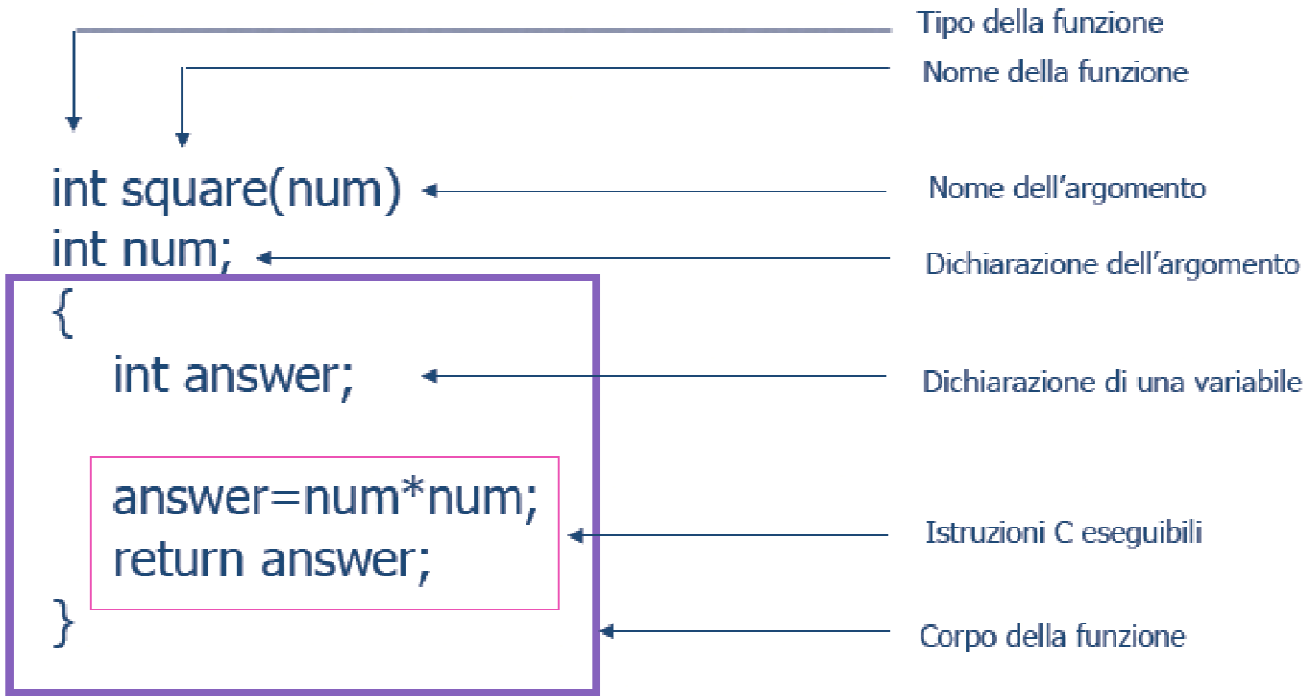
```c
#include <stdio.h>

int main()
{
    int a = 10, b = 20;
    int c = a * b;
    printf("a = %d b = %d c = %d\n", a, b, c);
    printf("modifico il valore di a\n");
    /* Ormai il valore di c è definito:
              se cambio a c non cambia! */
    a = 5;
    printf("a = %d b = %d c = %d\n", a, b, c);
    return 0;
}
```

Funzioni

- Una funzione C è costituita da un insieme di istruzioni del linguaggio C
- *Gerarchia di componenti*: definiamo strutture complesse a partire da componenti semplici
- Una funzione accetta dati in ingresso, li elabora e restituisce i risultati all'esterno

Esempio di funzione

- La funzione *square() accetta un* numero come dato in ingresso e restituisce il quadrato del numero come risultato;



La funzione **main**

- Ogni programma ne contiene una
- Indica il punto da cui iniziare l'esecuzione
- Rispetto alle altre funzioni si può non specificare il tipo della funzione e non vengono (di solito) dichiarati argomenti
- Può richiamare altre funzioni

```
int main()
{
    int n = 10;
    printf("Il quadrato di %d e' %d\n", n, square(n));
    return 0;
}
```

quadrato-f.c

- Creare cartella **lez5** nella vostra home
- Copiarvi il file **lez4/quadrato.c**
- Modificarlo per creare un programma che fa la stessa cosa ma utilizzando le funzioni
- *Copia/incolla codice*

```c
#include <stdio.h>

/* Questa funzione restituisce
 * il quadrato del suo argomento
 */
int square(int num)
{
    int answer;
    answer=num*num;
    return answer;
}

int main()
{
   int n = 10;
   printf("Il quadrato di %d e' %d\n",
          n, square(n));
   return 0;
}
```

La funzione scanf()

- La funzione scanf (definita in stdio.h) consente di leggere interattivamente l'input immesso dall'utente di un programma
- La sintassi sfrutta il passaggio di parametri per riferimento
- La funzione ha una sintassi simile a printf:

```
int scanf (const char* template , ...) ;
```

  - template è una stringa che specifica che cosa si vuole leggere
  - ... sta per un numero variabile di puntatori alle variabili in cui vogliamo salvare i valori letti
- La funzione ritorna il numero di assegnamenti (valori salvati in variabili) effettuati

scanf() – attenzione!

- Ogni volta che usate scanf per leggere un valore, ci deve essere memoria allocata per salvare il valore!

```
int p;
scanf ("%d", &p);
```

  - Si dichiara una variabile avente il tipo che intendiamo leggere
  - Si passa a scanf l'indirizzo di quella variabile **con &**

- In questo modo, abbiamo allocato spazio per salvare un dato di tipo int, e alla funzione diciamo a che indirizzo quello spazio si trova!

Esempio scanf() - quadrato-f2.c

- Chiedi all'utente di immettere un numero e calcolane il quadrato usando la funzione square.
- Modifica una copia di quadrato-f.c

```
int main()
{
    int n;
    printf("Inserisci un numero: ");
    scanf("%d",&n);
    printf("Il suo quadrato di %d\n", square(n));
    return 0;
}
```

Il costrutto if

Sintassi

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

- Esegui `blocco_if` solo se `<condizione> != 0`
- Se presente, esegui `blocco_else` solo se `<condizione> == 0`

Il costrutto if (cont)

- Blocchi `if` possono essere combinati per gestire piú di due casi, es:

```
int x = qualche_funzione(), y = altra_funzione();
if (x == 0) {
    //eseguito solo se x == 0
} else if (x > 5) {
    //eseguito solo se x > 5
} else if (x > 0) {
    //eseguito solo se x > 0 e x <= 5
} else {
    //eseguito in tutti i restanti casi (x < 0)
}
```

Il costrutto if (cont)

- L'ordine é importante!
- Confronta questo esempio con il precedente:

```
int x = qualche_funzione(), y = altra_funzione();
if (x == 0) {
    //eseguito solo se x == 0
} else if (x > 0) {
    //eseguito solo se x > 0
} else if (x > 5) {
    //eseguito solo se x > 5 e x <= 0, cioé MAI!!!
} else {
    //eseguito in tutti i restanti casi (x < 0)
}
```

Esempio: diff.c (tra due interi)

```c
#include<stdio.h>

int main()
{
    //Dichiarazione delle variabili
    int a, b, c;

    //Inserimento dati
    printf("Inserire primo intero: \n");
    scanf("%d", &a);
    printf("Inserire secondo intero: \n");
    scanf("%d", &b);

    //Determinazione del numero maggiore
    if (a>b){
        c=a-b;
        //Stampa risultato
        printf("Risultato di %d - %d: %d\n", a, b, c);
             }
    else{
        c=b-a;
        //Stampa risultato
        printf("Risultato di %d - %d: %d\n", b, a, c);
           }
    return 0;
}
```

Attenzione: errori comuni

- Nella scanf() l'& ci va. Cosa succede altrimenti?
  - `int x;`
  - `scanf("%d",x);`
- La parte sinistra di un assegnamento (L-value) rappresenta un luogo dove memorizzare un valore, non può essere un valore. Provate con…
  - `b-a = 100;`
- Il main è una funzione obbligatoria per ogni programma. Se non è presente una funzione main (cambiatele nome) che errore ricevete?