Variabili

Marco Alberti

Dipartimento
di Matematica
e Informatica

Università
degli Studi
di Ferrara

Programmazione e Laboratorio, A.A. 2024-2025

Ultima modifica: 6 dicembre 2023

Sommario

Premessa: commenti in C

In C un commento è il testo racchiuso

- fra la stringa `//` e la fine della linea (commento monolinea)
- fra la stringa `/*` e la successiva stringa `*/`, anche su linee diverse (commento multilinea)

I commenti sono ignorati dal compilatore; sono utili per annotare programmi.

040_variabili/commenti.c

```
1  #include <stdio.h>
2  main() {  // Un commento monolinea
3    printf("Hello,");
4    /* Un commento
5
6    multilinea */ printf(" World!\n");
7  }
```

Sommario

Modifica programma per funzionalità diverse

Abbiamo visto l'uso della macchina astratta C come calcolatrice.
Per calcolare il quoziente e il resto della divisione intera di 22 per 7:

040_variabili/quoziente-resto-costanti.c

```
1  #include <stdio.h>
2
3  main() {
4    printf("Quoziente: %d\n", 22 / 7);
5    printf("Resto: %d\n", 22 % 7);
6  }
```

Se volessi fare gli stessi calcoli, ma con 20 come dividendo, dovrei modificare il programma in tutti i punti in cui compare 22, sostituendolo con 20.
Processo (in generale) laborioso e soggetto a errori (potrei ad esempio dimenticare una occorrenza di 22).

Variabile

Alternativa:

4 Diamo un nome o identificatore (`dividendo`) e un tipo (`int`, intero) a una variabile che rappresenta il dividendo;

5 Assegniamo alla variabile il valore 22;

6 e 7 Usiamo il nome dove compariva il valore 22.

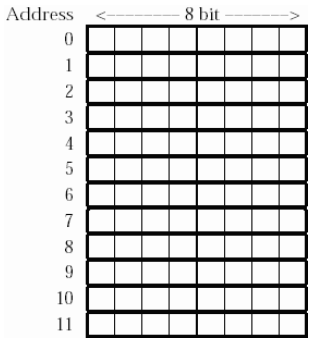
In questo modo, se vogliamo usare un altro valore abbiamo una sola modifica da fare. Quale?

040_variabili/quoziente-resto-variabile.c

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    dividendo = 22;
6    printf("Quoz.: %d\n", dividendo / 7);
7    printf("Resto: %d\n", dividendo % 7);
8  }
```

Memoria e variabili

- Assegnare un valore (ad esempio 22) a una variabile (ad esempio `dividendo`) significa scrivere quel valore in una cella di memoria.
- Usare direttamente gli indirizzi ("scrivi 22 all'indirizzo 7") sarebbe scomodo: difficili da ricordare e leggere, dipendente dall'hardware.
- Le variabili sono un'astrazione dell'area di memoria: anziché l'indirizzo si usa un identificatore indicativo.
- E' la macchina astratta ad associare l'identificatore all'indirizzo, in modo trasparente al programmatore.

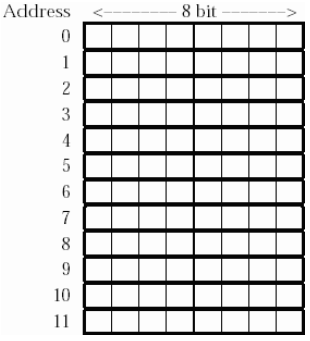


Definizione di variabile

Sintassi

⟨*definizione*⟩ ::= ⟨*tipo*⟩ ⟨*identificatore*⟩ **;**

Riserva un'area di memoria sufficiente a contenere un valore di tipo ⟨*tipo*⟩ e la etichetta con il nome ⟨*identificatore*⟩.



Sintassi identificatori

Sintassi

$\langle \textit{identificatore} \rangle ::= [\![\langle \textit{lettera} \rangle \mid \_]\!] [\![\langle \textit{lettera} \rangle \mid \langle \textit{cifra} \rangle \mid \_]\!]^*$

Un identificatore deve iniziare con una lettera o un underscore, e continuare con zero o più lettere, cifre o underscore.

Esercizio

Quali delle seguenti stringhe sono identificatori validi?

- `a`
- `_`
- `1`
- `nome1`
- `nome_1_v`
- `_a1`
- `a___`

Espressioni con variabili

Come al solito, nei programmi le variabili si utilizzano per mezzo di opportune espressioni.

Sintassi (provvisoria)

$\langle espressione \rangle ::= \langle espressioneDiOutput \rangle$
$\quad | \quad \langle espressioneIntera \rangle$
$\quad | \quad \langle espressioneAssegnamento \rangle$
$\quad | \quad \langle espressioneVariabile \rangle$

Espressione variabile

Consente di riferirsi al valore di una variabile in un'espressione.

Sintassi

⟨*espressioneVariabile*⟩ ::= ⟨*identificatore*⟩

Semantica

- Effetto: nessuno
- Valore: il valore dell'area di memoria identificata da ⟨*identificatore*⟩

Esempio

Nel programma alla slide 3, l'espressione `dividendo / 7` contiene l'espressione variabile `dividendo`.
Se l'area di memoria corrispondente a `dividendo` contiene il valore `22`, allora `dividendo` → `22` e `dividendo / 7` → `3`

Espressione di assegnamento

Serve ad assegnare un valore a una variabile, cioè a scrivere nell'area di memoria corrispondente.

Sintassi

$\langle espressioneAssegnamento \rangle ::= \langle lValue \rangle$ **=** $\langle espressione \rangle$

$\langle lValue \rangle ::= \langle identificatore \rangle$

Semantica

- Effetto: il valore di $\langle espressione \rangle$ viene scritto nell'area identificata da $\langle lValue \rangle$
- Valore: il nuovo valore dell'area identificata da $\langle lValue \rangle$

Esercizio

Doppio

Scrivere un programma che

1. Definisca una variabile intera di nome `v`;
2. Assegni a `v` il valore 4;
3. Scriva in output il doppio di `v` (calcolandolo).

L'assegnamento è distruttivo

Che cosa stampa il seguente programma?

040_variabili/assegnamento-distruttivo.c

```
1  #include <stdio.h>
2
3  main() {
4    int a;
5    a = 2;
6    printf("%d\n", a);
7    a = 3;
8    printf("%d\n", a);
9  }
```

E' possibile assegnare successivamente valori diversi alla stessa variabile. Il nuovo valore assegnato sovrascrive quello precedente, cioè va a occupare l'area di memoria della variabile, cancellando il contenuto precedente.

Variabili non inizializzate

Attenzione

Prima del primo assegnamento, il valore di una variabile è, in generale, imprevedibile.

Che cosa stampa il seguente programma?

040_variabili/non-inizializzata.c

```
1  #include <stdio.h>
2
3  main() {
4    int a;
5    // manca inizializzazione di a!
6    printf("%d\n", a);
7  }
```

Le variabili non devono essere usate (cioè riferite in espressioni) prima di essere inizializzate.

Catene di assegnamenti

L'operatore di assegnamento è associativo a destra.

Esempio

Supponendo che `a`, `b` e `c` siano variabili intere, l'espressione `a = b = c = 1` equivale a `a = (b = (c = 1))`

Quanto valgono `a`, `b` e `c` dopo la valutazione di questa espressione?

Sommario

Stato della macchina astratta

Lo stato della macchina astratta è quell'insieme di informazioni che consente di prevedere l'effetto dell'esecuzione del programma da parte della macchina astratta fino alla fine.

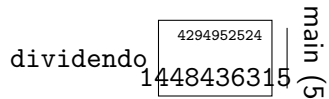
Consiste in

- stato della memoria (valore in tutte le celle di memoria)
- posizione nel programma (prossima istruzione che sarà eseguita)

L'effetto di un'istruzione è una modifica dello stato; saper leggere un'istruzione significa (anche) saper prevedere, dato lo stato attuale, il nuovo stato conseguente all'esecuzione dell'istruzione.

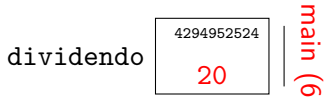
Quoziente e resto con variabile

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    dividendo = 20;
6    printf("Quoz.: %d\n", dividendo / 7);
7    printf("Resto: %d\n", dividendo % 7);
8  }
```



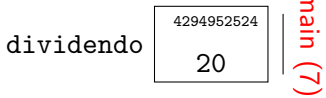
Quoziente e resto con variabile

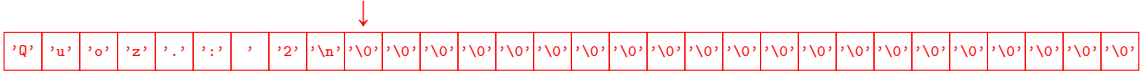
```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    dividendo = 20;
6    printf("Quoz.: %d\n", dividendo / 7);
7    printf("Resto: %d\n", dividendo % 7);
8  }
```



Quoziente e resto con variabile

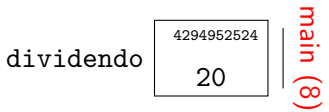
```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    dividendo = 20;
6    printf("Quoz.: %d\n", dividendo / 7);
7    printf("Resto: %d\n", dividendo % 7);
8  }
```





Quoziente e resto con variabile

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    dividendo = 20;
6    printf("Quoz.: %d\n", dividendo / 7);
7    printf("Resto: %d\n", dividendo % 7);
8  }
```



Sommario

Input

Il programma che calcola quoziente e resto sarebbe più utile se si potesse eseguire con dividendi diversi senza dover modificarlo e ricompilarlo.
Il modo c'è: anziché assegnare alla variabile `dividendo` un valore costante, chiediamo all'utente il valore del dividendo (operazione di input) e lo scriviamo nella variabile, dopo di che procediamo come prima.
L'operazione di input, come l'operazione di output, richiede un'espressione apposita.

Input

Sintassi

⟨*espressione*⟩ ::= ⟨*espressioneDiOutput*⟩
| ⟨*espressioneIntera*⟩
| ⟨*espressioneAssegnamento*⟩
| ⟨*espressioneVariabile*⟩
| ⟨*espressioneDiInput*⟩

⟨*espressioneDiInput*⟩ ::=
`scanf("` ⟦ ⟨*specificatoreConversione*⟩ ⟧$^+$ `"` ⟦ `,` ⟨*indirizzo*⟩ ⟧$^+$ `)`

⟨*specificatoreConversione*⟩ ::= `%d`

⟨*indirizzo*⟩ ::= `&` ⟨*identificatore*⟩

Il numero degli specificatori di conversione e quello degli indirizzi devono coincidere; determinano il numero di valori da leggere da input.

Esempio

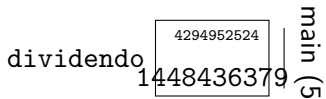
040_variabili/quoziente-resto-input.c

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    printf("Inserisci un numero intero\n");
6    scanf("%d", &dividendo);
7    printf("Q: %d\n", dividendo / 7);
8    printf("R: %d\n", dividendo % 7);
9  }
```

L'espressione `scanf("%d", &dividendo)` legge da tastiera un numero intero (come indicato dallo specificatore di conversione `%d`, lo stesso utilizzato per l'output di interi) e lo scrive all'indirizzo corrispondente alla variabile `dividendo` (l'effetto è lo stesso dell'assegnamento alla variabile `dividendo`).

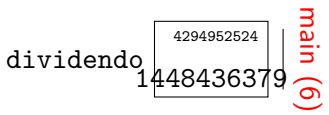
Quoziente e resto con input

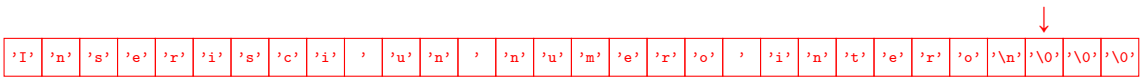
```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    printf("Inserisci un numero intero\n");
6    scanf("%d", &dividendo);
7    printf("Q: %d\n", dividendo / 7);
8    printf("R: %d\n", dividendo % 7);
9  }
```



Quoziente e resto con input

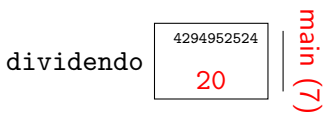
```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    printf("Inserisci un numero intero\n");
6    scanf("%d", &dividendo);
7    printf("Q: %d\n", dividendo / 7);
8    printf("R: %d\n", dividendo % 7);
9  }
```

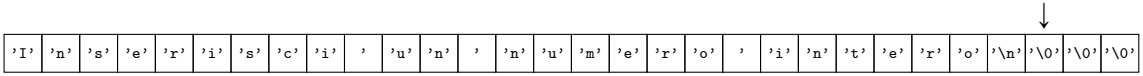




Quoziente e resto con input

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo;
5    printf("Inserisci un numero intero\n");
6    scanf("%d", &dividendo);
7    printf("Q: %d\n", dividendo / 7);
8    printf("R: %d\n", dividendo % 7);
9  }
```





Quoziente e resto con input

```
#include <stdio.h>

main() {
  int dividendo;
  printf("Inserisci un numero intero\n");
  scanf("%d", &dividendo);
  printf("Q: %d\n", dividendo / 7);
  printf("R: %d\n", dividendo % 7);
}
```

main (8)

dividendo	4294952524 20

↑

'2'	'0'	'\n'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'

stdin

'I'	'n'	's'	'e'	'r'	'i'	's'	'c'	'i'	' '	'u'	'n'	' '	'n'	'u'	'm'	'e'	'r'	'o'	' '	'i'	'n'	't'	'e'	'r'	'o'	'\n'	'Q'	':'	' '

stdout

Esercizio

Doppio con variabile

Scrivere un programma che

1. definisca una variabile intera di nome `v`;
2. richieda all'utente un valore per `v`;
3. stampi il doppio di `v`.

Convincersi della correttezza del programma provandolo con vari valori di input.

Abbreviazioni

Per praticità, il linguaggio C consente, in una sola definizione, di:

- definire più variabili dello stesso tipo, separandone gli identificatori con una virgola (ad esempio
  `int a, b;`
  definisce le due variabili `a` e `b` di tipo `int`)
- inizializzare le variabili con un valore (ad esempio
  `int a = 10;`
  definisce la variabile di tipo `int a` e assegna ad `a` il valore `10`)
- entrambe le cose (`int a = 5, b = 10;`)
- definire diverse variabili ed inizializzarne solo alcune (ad esempio `int a = 5, b, c = 10;` definisce le variabili `a`, `b` e `c` di tipo `int` e inizializza `a` a `5` e `c` a `10`, lasciando `b` non inizializzata.

Due variabili

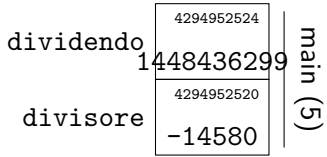
Abbiamo definito una variabile contenente il dividendo. Ovviamente possiamo usare un'altra variabile per il divisore.

040_variabili/quoziente-resto-due-var.c

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo, divisore;
5    scanf("%d%d", &dividendo, &divisore);
6    printf("Q: %d\n", dividendo / divisore);
7    printf("R: %d\n", dividendo % divisore);
8  }
```

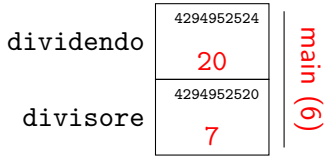
Quoziente e resto, due variabili

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo, divisore;
5    scanf("%d%d", &dividendo, &divisore);
6    printf("Q: %d\n", dividendo / divisore);
7    printf("R: %d\n", dividendo % divisore);
8  }
```



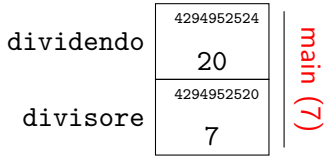
Quoziente e resto, due variabili

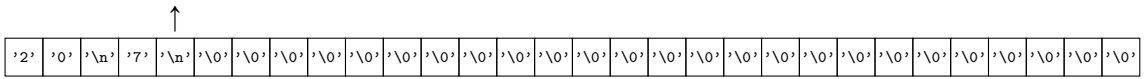
```
1  #include <stdio.h>
2
3  main() {
4    int dividendo, divisore;
5    scanf("%d%d", &dividendo, &divisore);
6    printf("Q: %d\n", dividendo / divisore);
7    printf("R: %d\n", dividendo % divisore);
8  }
```

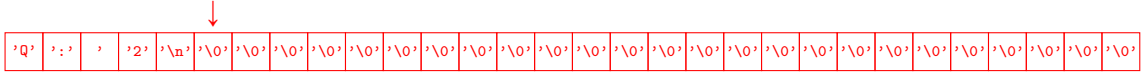


Quoziente e resto, due variabili

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo, divisore;
5    scanf("%d%d", &dividendo, &divisore);
6    printf("Q: %d\n", dividendo / divisore);
7    printf("R: %d\n", dividendo % divisore);
8  }
```

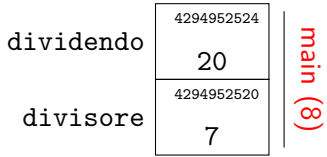


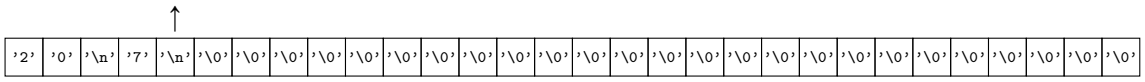


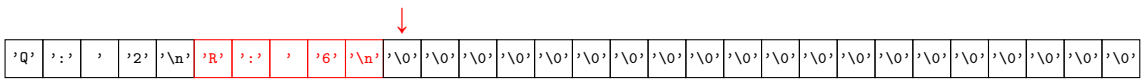


Quoziente e resto, due variabili

```
1  #include <stdio.h>
2
3  main() {
4    int dividendo, divisore;
5    scanf("%d%d", &dividendo, &divisore);
6    printf("Q: %d\n", dividendo / divisore);
7    printf("R: %d\n", dividendo % divisore);
8  }
```







Esercizio

Somma

Calcolare la somma di tre numeri richiesti all'utente.

Esercizio

Quadrato

Calcolare l'area e il perimetro di un quadrato il cui lato è richiesto all'utente.

Esercizio

Rettangolo

Calcolare l'area e il perimetro di un rettangolo richiedendo all'utente la base e l'altezza

Sommario

Aggiornamento di una variabile

L'espressione `a = a + 1`, letta come equazione, non ha soluzione.
In C invece è un'espressione valida.
Supponendo che `a` sia una variabile intera di valore `8` prima della valutazione dell'espressione:

1. viene valutata l'espressione a destra dell'`=`: `a + 1` $\rightarrow$ `9`
2. il valore viene scritto nella variabile `a`

Poiché il lato destro dell'assegnamento viene valutato prima della scrittura in memoria, la `a` a destra dell'`=` vale il valore di `a` prima dell'assegnamento.
L'effetto è di incrementare di `1` il valore di `a`.

Esercizio

Se a è una variabile intera, scrivere espressioni che la aggiornino

- al suo valore incrementato di 2
- al doppio del suo valore
- al quadrato del suo valore

Verificare la correttezza delle espressioni con opportuni programmi che

1. Richiedano in input il valore a;
2. Aggiornino a come richiesto;
3. Stampino in output il nuovo valore di a;

Assegnamento compatto

L'aggiornamento di una variabile (cioè l'assegnamento ad essa di un valore dipendente dal valore attuale) è così frequente che il linguaggio offre versioni abbreviate:

Sintassi

⟨*asseghamentoCompatto*⟩ ::= ⟨*lValue*⟩ ⟨*op*⟩**=** ⟨*espressione*⟩

⟨*op*⟩ ::= **+** | **-** | ***** | **/** | **%**

Semantica

- Effetto: a ⟨*lValue*⟩ viene assegnato il valore dell'espressione ⟨*lValue*⟩ ⟨*op*⟩ ⟨*espressione*⟩
- Valore: il nuovo valore di ⟨*lValue*⟩

Esempio

`a += 2` equivale ad `a = a + 2`

Esercizio

Abbreviare le espressioni dell'esercizio alla slide 25.

Operatori di incremento e decremento

Il caso più frequente di aggiornamento è l'incremento o decremento di 1; il C offre operatori appositi, che possono essere prefissi (l'operatore precede l'lValue) o postfissi (l'operatore segue l'lValue).

Sintassi

$\langle incremento \rangle ::= \langle lValue \rangle \texttt{++} \mid \texttt{++} \langle lValue \rangle$

$\langle decremento \rangle ::= \langle lValue \rangle \texttt{--} \mid \texttt{--} \langle lValue \rangle$

Semantica

- Effetto: incrementa o decrementa $\langle lValue \rangle$ di 1
- Valore: il valore di $\langle lValue \rangle$ prima dell'incremento o decremento per gli operatori postfissi; il valore dopo l'incremento o decremento per gli operatori prefissi

Operatori di incremento e decremento

Esempio

Se `a` è una variabile intera di valore `8`:

Espressione	Effetto: `a` vale	Valore
`a++`	9	8
`++a`	9	9
`a--`	7	8
`--a`	7	7

Aggiornamento tabella priorità e associatività

Famiglia	Operatore	Priorità	Associatività
Incr. e decr. postfissi	++, --	1	Sinistra
Unari Incr. e decr. prefissi	+,- ++, --	2	Destra
Binari Moltiplicativi	*,/,%	3	Sinistra
Binari Additivi	+,-	4	Sinistra
Assegnamento Assegnamento compatto	= +=,-=,*=,/=,%=	14	Destra

Esercizio

Se `a` è una variabile intera di valore `5`, quali sono valore ed effetto delle seguenti espressioni?

- `a += a + 6`
- `a += a = 4`
- `a += a++`
- `a + a++`