LINGUAGGIO C: Input/Output e costrutti

Fondamenti di Programmazione 2021/2022

Francesco Tortorella

Istruzioni di calcolo e assegnazione

- L'effetto è di aggiornare il valore di una variabile di un certo tipo con il valore ottenuto dalla valutazione di un'espressione dello stesso tipo.
- Il formato è:

```
variabile = espressione;
```

left_value = right_value

- Esempi:

```
a=4;        a=a+1;  cond = x > y;
b=0;        a=a+b;  cond =(a>=0)&&(a<=9);
b=a;
```

Istruzioni di calcolo e assegnazione

- Quali sono istruzioni corrette?

```
int i, j, val_m;
const int ci = i;

2040 = val_m;
i + j = val_m;
ci = val_m;
i = j;
```

Assegnazione composta

op	uso	equivale a	descrizione
=	`a = b`		assegna il valore di *b* alla variabile *a*
+=	`a += b`	`a = a + b;`	somma *a* e *b* ed assegna il risultato alla variabile *a*
-=	`a -= b`	`a = a - b;`	sottrae *b* ad *a* ed assegna il risultato alla variabile *a*
***=**	`a *= b`	`a = a * b;`	moltiplica *a* per *b* ed assegna il risultato alla variabile *a*
/=	`a /= b`	`a = a / b;`	divide *a* per *b* ed assegna il risultato alla variabile *a*
%=	`a %= b`	`a = a % b;`	mette in *a* il resto della divisione intera di *a* per *b*

Autoincremento e autodecremento

istruzione	istruzione equivalente	restituisce	
`++x;`	`x=x+1;`	variabile	preincremento
`x++;`	`x=x+1;`	valore	postincremento
`--x;`	`x=x-1;`	variabile	predecremento
`x--;`	`x=x-1;`	valore	postdecremento
`y=++x;`	`x=x+1;y=x;`		
`y=x++;`	`y=x;x=x+1;`		

Operazioni di Input/Output

- Con le operazioni di **input**, il valore di una variabile viene modificato con il valore ottenuto grazie ad un'operazione di lettura dall'unità di ingresso (tastiera).

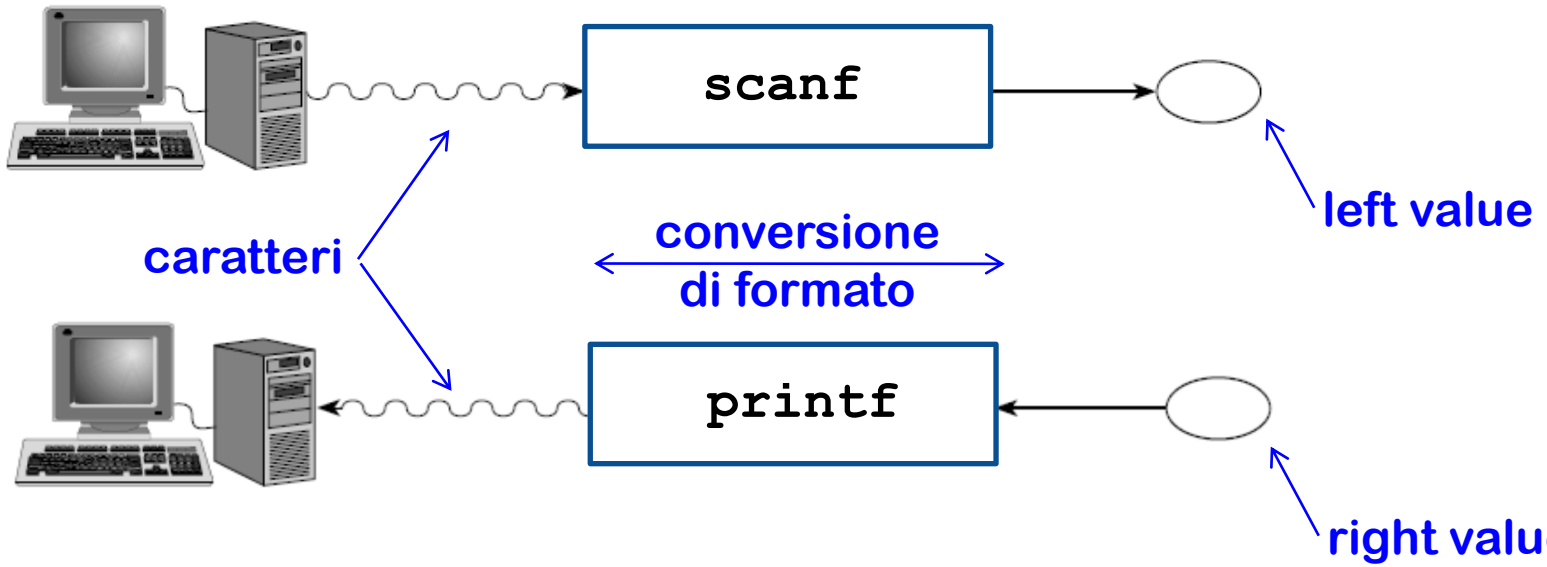
- Con le operazioni di **output**, un'espressione viene valutata ed il valore ottenuto viene presentato sull'unità di uscita (schermo).

Operazioni di Input/Output

- Il C usa un'utile astrazione (i **flussi** o **streams**) per realizzare le operazioni di I/O con dispositivi come la tastiera e lo schermo.
- Uno stream è un oggetto dove un programma può inserire o estrarre caratteri e che virtualizza i dispositivi fisici ad esso associati.
- La libreria standard C include il file header **`stdio.h`** dove sono dichiarate le funzioni per input e output.

Operazioni di Input/Output

- Gli stream gestiscono flussi di caratteri
- L'operazione di input avviene su una variabile (left value)
- L'operazione di output avviene su un'espressione (right value)



Operazione di output: `printf`

`printf``(`*`stringa di formato, altri argomenti`*`)`

- Stampa sullo standard output i caratteri contenuti nella *stringa di formato*.
- Se la stringa contiene **specificatori di formato** (sequenze di caratteri inizianti con %), gli argomenti addizionali sono *formattati* e inseriti nella stringa in output al posto dei loro rispettivi specificatori.
- Il numero degli argomenti deve essere uguale a quello degli specificatori.
- Per utilizzare **`printf`** bisogna introdurre **`#include <stdio.h>`**

Operazione di output: `printf`

- Alcuni specificatori di formato:
- **`%d`** per gli interi (**`int`**)
- **`%f`** per i numeri reali (**`float`** e **`double`**)
- **`%c`** per i caratteri singoli (**`char`**)
- **`%s`** per le stringhe di caratteri

Esempio

```
#include <stdio.h>
int main() {
    int p;
    p = 18;
    printf("Io ho %d anni\n", p);
}
```

Operazione di input: `scanf`

`scanf`(*`stringa di formato`*,*`elenco variabili`*)

- La funzione **`scanf`** legge caratteri dallo stream di input, li converte in dati di un certo tipo in accordo a quanto precisato dalla *stringa di formato*. I dati sono poi memorizzati all'interno delle variabili presenti *nell'elenco variabili*.

Operazione di input: `scanf`

- La *stringa di formato* è simile a quanto visto per **printf**
- Contiene:
  - ***Caratteri blank***, in corrispondenza dei quali la funzione leggerà e salterà tutti i caratteri blank che incontra in input prima del primo carattere non-blank. Sono caratteri blank lo spazio, il newline ed il carattere di tabulazione;
  - ***Specificatori di formato***: una sequenza formata da un carattere iniziale %, usata per specificare il tipo ed il formato del dato da leggere in input e da assegnare alla variabile corrispondente.
- Ci dovranno essere tanti specificatori di formato quante sono le variabili nell'elenco.

Operazione di input: `scanf`

- Alcuni specificatori di formato:
- `%d` – legge una sequenza di caratteri costituita da cifre decimali (0-9), opzionalmente preceduta da un segno (+ o -). La lettura della sequenza termina al primo carattere blank. Il dato viene memorizzato come un intero nella variabile corrispondente
- `%c` – legge il prossimo carattere e lo salva nella variabile corrispondente
- `%f` - legge una sequenza di caratteri costituita da cifre decimali (0-9), opzionalmente contenente un punto decimale, opzionalmente preceduta da un segno (+ o -). La lettura della sequenza termina al primo carattere blank. Il dato viene memorizzato come un reale nella variabile corrispondente

Operazione di input: `scanf`

- L'elenco delle variabili contiene gli identificatori delle variabili che saranno assegnate dalla funzione **`scanf`**
- Più precisamente, di ogni variabile sarà fornito l'indirizzo di memoria nel quale essa è memorizzata. A questo scopo, si utilizza l'operatore & che viene prefisso ad ogni variabile da modificare.
- Esempio (lettura da input di due variabili intere):

```
int a,b;
scanf("%d %d", &a, &b);
```

Strutture di controllo

- Ricordiamo che ogni algoritmo può essere implementato impiegando solo tre tipi di strutture per il controllo di flusso:
  - **Sequenza**
  - **Selezione**
  - **Ciclo**
- Tipicamente i linguaggi offrono più di un costrutto per ogni tipo di struttura per rendere più agevole la codifica degli algoritmi

Sequenza

- La sequenza è costituita da un insieme di istruzioni successive che vengono eseguite nell'ordine in cui compaiono nel testo.
- Un particolare caso di sequenza in C è il **blocco** (o *istruzione composta*, *compound statement*), formato da un insieme di istruzioni tra parentesi graffe { }.

Costrutti di selezione

- Permettono di scegliere di eseguire una tra due istruzioni alternative in base alla valutazione di una espressione logica
- Tre principali costrutti
  - `if`
  - `if … else`
  - `if … else if … else`

Costrutti di selezione: `if`

Sintassi

```
if(condizione)
  istruzione
```

- L'*istruzione* è eseguita solo se *condizione* è **true**
- L'*istruzione* può essere costituita da un blocco.

Esempi

- Calcolo del valore assoluto di un numero dato in input
- Verificare che due valori X e Y forniti in input rispettino la condizione X >= Y.

Costrutti di selezione: `if…else`

Sintassi

```
if(condizione)
  istruzione_1
else
  istruzione_2
```

- L'*istruzione_1* è eseguita solo se *condizione* è **true**
- L'*istruzione_2* è eseguita solo se *condizione* è **false**
- *istruzione_1* e *istruzione_2* possono essere costituite da blocchi

Esempio: max fra due

```
# include <stdio.h>

int main()
{
  int x,y,max;

  printf("Primo valore: ");
  scanf("%d",&x);
  printf("Secondo valore: ");
  scanf("%d",&y);

  if(x>y)
    max=x;
  else
    max=y;

  printf("Il massimo tra %d e %d e': %d\n", x, y, max);
}
```

Esempio: max fra tre

```
# include <stdio.h>

int main()
{
  int x,y,z,max;

  printf("Primo valore: ");
  scanf("%d",&x);
  printf("Secondo valore: ");
  scanf("%d",&y);
  printf("Terzo valore: ");
  scanf("%d",&z);

  max=x;

  if(y>max)
    max=y;

  if(z>max)
    max=z;

  printf("Il massimo e': %d\n", max);
}
```

Costrutti di selezione: `if…else if … else`

Sintassi

```
if(condizione_1)
  istruzione_1
else if(condizione_2)
  istruzione_2
else if(condizione_3)
  istruzione_3
else
  istruzione_4
```

- L'*istruzione_1* è eseguita solo se *condizione_1* è **true**
- L'*istruzione_2* è eseguita solo se *condizione_1* è **false** e *condizione_2* è **true**
- L'*istruzione_3* è eseguita solo se *condizione_1* è **false**, *condizione_2* è **false** e *condizione_3* è **true**
- L'*istruzione_4* è eseguita solo se *condizione_1* è **false**, *condizione_2* è **false** e *condizione_3* è **false**
- *istruzione_1* e *istruzione_2* possono essere costituite da blocchi

Esempio

```
# include <stdio.h>

int main()
{
  int voto;

  printf("Voto ricevuto: ");
  scanf("%d", &voto);

  if (voto < 18)
   printf(" Ritorna\n");
  else if (voto < 24)
    printf(" Si puo' dare di piu'! \n ");
  else if (voto < 27)
    printf(" Non c'e' male !\n ");
  else if (voto < 30)
    printf("Bene!\n");
  else if (voto == 30)
    printf(" Finalmente ci siamo ! \n");
  else
    printf(" WOW!!! \n");
}
```

Costrutti di ciclo

- Servono a ripetere l'esecuzione di un'istruzione
- A seconda di come viene definito il numero di ripetizioni dell'esecuzione, si distinguono in
  - **Costrutti di ciclo a condizione**
  - **Costrutti di ciclo a conteggio**

Costrutti di ciclo: `while`

- E' un costrutto di ciclo **a condizione iniziale**
- Non si definisce esplicitamente il numero di ripetizioni dell'esecuzione, ma si valuta all'inizio del ciclo un'espressione logica che, fin quando risulta vera, causa un'ulteriore esecuzione dell'istruzione.

Costrutti di ciclo: `while`

Sintassi

```
while(condizione)
  istruzione
```

- Si valuta la *condizione*
- Se risulta vera, si esegue l'istruzione e quindi si torna a verificare la condizione
- Se la condizione risulta falsa, si passa a eseguire le istruzioni che si trovano dopo la chiusura del **`while`**
- Qual è il minor numero di cicli che si può effettuare ?

Costrutti di ciclo: `do...while`

- E' un costrutto di ciclo **a condizione finale**
- Non si definisce esplicitamente il numero di ripetizioni dell'esecuzione, ma si valuta al termine del ciclo un'espressione logica che, fin quando risulta vera, causa un'ulteriore esecuzione dell'istruzione.

Costrutti di ciclo: **`do...while`**

Sintassi

```
do
    istruzione
while(condizione);
```

- Si esegue l'*`istruzione`*
- Si valuta la *`condizione`*
- Se risulta vera, si torna ad eseguire l'*`istruzione`*
- Se la condizione risulta falsa, si passa a eseguire le istruzioni che si trovano dopo la chiusura del **`while`**
- Qual è il minor numero di cicli che si può effettuare ?

Costrutti di ciclo: `for`

- E’ un costrutto di ciclo **a conteggio**
- Si definisce esplicitamente il numero di ripetizioni dell’esecuzione
- Il conteggio viene gestito grazie ad una variabile (*variabile di conteggio*) che assume un valore iniziale e viene incrementata di un valore fisso ad ogni ripetizione del ciclo finché non raggiunge o supera un valore finale.

Costrutti di ciclo: `for`

Sintassi

```
for(inizialization; condition; increase)
  istruzione
```

- Si esegue *inizialization*
- Si valuta *condition*
- Se risulta vera, si esegue l'*istruzione;* al termine dell'esecuzione, si esegue *increase* e si torna a valutare *condition*
- Se la condizione risulta falsa, si passa a eseguire le istruzioni che si trovano dopo la chiusura del **for**

Problema

- Leggere da input un insieme di numeri interi e calcolarne la media, il valore minimo ed il valore massimo.
- Il numero di valori da leggere è fornito in ingresso prima della sequenza di valori