Programmazione (imperativa)

Corso di Laurea in Informatica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni:	Lunedì 12.00 - 13.00 e 14.00 - 16.00	Mercoledì 14.00 - 17.00
Laboratorio:	Giovedì 12.00 - 13.00 e 14.00 - 17.00	
Ricevimento:	su appuntamento	
Tel.:	02 503 16235	
E-mail:	roberto.cordone@unimi.it	
Web page:	http://homes.di.unimi.it/~cordone/courses/2015-prog/2015-prog.html	

Parsing — Milano, A.A. 2015/16

Parsing

Parsing (analisi sintattica) è il processo che analizza una data sequenza di simboli per determinarne la struttura grammaticale

Questa operazione è fondamentale nell'esecuzione di programmi

- lettura da tastiera dei dati e loro interpretazione
- interpretazione di stringhe che contengono dati
- lettura da file dei dati e loro interpretazione

Stream in ingresso è la fonte dei dati (tastiera, stringa, file)

Interpretare lo stream in ingresso non è affatto elementare

Parser è un programma che esegue tale analisi

La funzione scanf

```
int scanf(char *formato, ...)
```

è la principale funzione per il parsing di testo fornito da tastiera

- ha un numero variabile di parametri ($p \geq 1$)
  - la stringa di formato definisce un *pattern* che la funzione cerca di riconoscere nello stream in ingresso corrente
  - $p - 1$ puntatori a oggetti (variabili, elementi di vettori, campi di strutture, blocchi di memoria) consentono di conservare i risultati dell'interpretazione
- termina alla fine della stringa o al primo oggetto non riconosciuto
- restituisce il numero di oggetti del *pattern* ritrovati nello stream

Il resto dello stream resta disponibile per la chiamata seguente

La stringa di formato

Stringa di formato e stream in ingresso sono compatibili quando

- ogni carattere non separatore nella stringa, cioè diverso da spazio, a capo e tabulazione, corrisponde allo stesso carattere nello stream
- uno o più separatori consecutivi nella stringa corrispondono a zero o più separatori consecutivi nello stream
- le specifiche di conversione nella stringa (`%...`) corrispondono a zero o più separatori consecutivi e una sequenza di caratteri che descrive un oggetto del tipo indicato dalla specifica (numero intero, numero reale, singolo carattere, parola, ...)

Esempio:

La stringa di formato "`Numero  %d/%d`" corrisponde allo stream "`Numero`" seguito da zero o più spazi e da due interi separati da '`/`'

Quindi è compatibile con

"`Numero3/5`", "`Numero 15/ 2`", "`Numero -6/ 0`", ecc...

Non è compatibile con "`Numero 10 /5`"

Specifica di conversione

Indica il tipo di oggetto atteso nella posizione corrente dell'ingresso

%d	`int` decimale
%u	`unsigned int` decimale
%f	`float` in notazione decimale
%e	`float` in notazione scientifica
%c	`char`
%s	stringa priva di separatori
%[*set*]	stringa fatta con i caratteri elencati in *set*

Ogni puntatore indica dove copiare l'oggetto corrispondente

L'istruzione `scanf("%d/%d/%d",&giorno,&mese,&anno);`

- cerca tre numeri interi separati da `'/'`
- li assegna alle variabili `giorno`, `mese` e `anno`
- restituisce il numero di interi trovati

Esempio

Ingresso	Istruzione	n	i	j
12 , 34	`n = scanf("%d%d",&i,&j);`	1	12	invariato
12 , 34	`n = scanf("%d,%d",&i,&j);`	1	12	invariato
12 , 34	`n = scanf("%d ,%d",&i,&j);`	2	12	34
12 , 34	`n = scanf("%d, %d",&i,&j);`	1	12	invariato
12 , 34	`n = scanf("%d , %d",&i,&j);`	2	12	34

Suggerimento: si noti la posizione della virgola nella stringa di formato

Raffinamenti

I modificatori di lunghezza alterano il tipo dell'oggetto

`%hd`	`short` decimale
`%hu`	`unsigned short` decimale
`%ld`	`long` decimale
`%lu`	`unsigned long` decimale
`%lf`	`double`
`%Lf`	`long double`

Un numero (larghezza massima) indica che l'oggetto trovato in ingresso deve contenere un massimo numero dato di caratteri: con ingresso "2010", l'istruzione `scanf("%2d",&i);` assegna a `i` il valore 20

Un asterisco (soppressore) indica che l'oggetto trovato in ingresso non va assegnato ad alcun puntatore: `scanf("%*d",&i);` non altera `i`

Esempio

Ingresso	Istruzione	Risultati
`12 34`	`n = scanf("%*d%d",&i);`	`n = 1` `i = 34`
`Un due tre`	`n = scanf("%*s%s",s);`	`n = 1` `s = "due"`

`12345`	`n = scanf("%1d%2d%3d",&i,&j,&k);`	`n = 3`
		`i = 1`
		`j = 23`
		`k = 45`

`123456`	`n = scanf("%2d%2s%2d",&i,s,&j);`	`n = 3`
		`i = 12`
		`s = "34"`
		`j = 56`

Si noti la mancanza della & prima di s

Esempio

Si può indicare una sequenza di caratteri consecutivi con gli estremi

- `[a-z]` indica tutte le lettere minuscole
- `[0-9ae]` indica tutte le cifre, più i caratteri `'a'` e `'b'`

Si indica un insieme di caratteri proibiti premettendo `'^'` al loro elenco

- `[^dgt]` indica tutte le stringhe che non contengono `'d'`, `'g'` e `'t'`

Ingresso	Istruzione	Risultati
`123abc`	`n = scanf("%[0-9]",s);`	`n = 1` `s = "123"`
`abc123`	`n = scanf("%[0-9]",s);`	`n = 0` `s` invariato
`abc123`	`n = scanf("%[^0-9]",s);`	`n = 1` `s = "abc"`

Si noti la mancanza della `&` prima di `s`

Uscita formattata

```
int printf(char *formato, ...)
```

è la principale funzione per la stampa a video

Funziona esattamente come la funzione `scanf`

Ha un numero variabile di parametri $p \geq 1$

- la stringa di formato definisce un *pattern* che la funzione ricostruisce nell'uscita
- $p - 1$ oggetti (variabili, elementi di vettori, campi di strutture, blocchi di memoria) dai quali la funzione trae i risultati da stampare

Restituisce il numero di oggetti del *pattern* stampati in uscita

Differenze tra `printf` e `scanf`

- i parametri sono passati per copia e quindi non sono puntatori (tranne le stringhe)
- i separatori non vengono ignorati, ma stampati esattamente
- la specificazione di larghezza diventa una larghezza minima:
  - è ignorata se l'oggetto è più grosso
  - è imposta con l'aggiunta di spazi se è più piccolo

Esempio:

`scanf("%2d",&i);` con ingresso da tastiera "2014" pone `i = 20`

`printf("%4d",i);` con `i = 20` scrive a video " 20" (due spazi bianchi)

Raffinamenti

I seguenti flag (subito dopo %) influenzano la stampa quando si è specificata una larghezza eccessiva per l'oggetto:

- `-` indica di allineare l'oggetto a sinistra anziché a destra (`"20 "`)
- `0` indica di completare la stampa con `0` anziché spazi (`"0020"`)

Nella stampa di numeri reali, si può specificare la precisione

`"%`*n*`.`*d*`lf"`

- *n* indica la larghezza minima totale
- *d* indica il numero di cifre decimali esatto

L'istruzione `printf("r:%8.3lf",30.5);`

stampa a video `"r:  30.500"` (due spazi bianchi fra `r` e il numero)

Parsing di stringhe

Tutti gli stream di ingresso sono gestiti con funzioni simili
(tastiera, stringhe e file)

```
int sscanf(char *stringa, char *formato, ...)
```

- riconosce nella *stringa* (primo parametro)
- gli elementi forniti dalla stringa di *formato* (secondo parametro)
- assegna gli oggetti riconosciuti ai puntatori (parametri successivi)

Esempio:

```
char s[10+1];
int giorno, mese, anno;

strcpy(s,"14/05/2010");
sscanf(s,"%d/%d/%d",&giorno,&mese,&anno);
```