CORSO DI PROGRAMMAZIONE
A.A. 2014-15

Dispensa 9

Laboratorio

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

Funzioni per i flussi

Un **flusso** è una sequenza di caratteri o meglio di byte di dati. Una sequenza di byte che entra in un programma è un flusso di input mentre una sequenza di byte che esce dal programma è un flusso di output. Concentrandosi sui flussi, si può fare a meno di preoccuparsi dell'origine o della destinazione dei dati; il vantaggio principale è che in questo modo la programmazione è indipendente dai dispositivi.

I flussi in C si dividono in due categorie: di testo e binari. Un **flusso di testo** è costituito soltanto da caratteri, come i dati inviati allo schermo. I flussi di testo sono strutturati in righe che possono essere lunghe fino a 255 caratteri e sono concluse da un carattere di nuova riga.

I **flussi binari** possono gestire dati di qualsiasi tipo, tra cui anche quelli di testo. I byte in un flusso binario non sono tradotti o interpretati in modo particolare ma letti semplicemente così come sono. I flussi binari vengono utilizzati principalmente per i file su disco.

L'ANSI C ha tre flussi predefiniti chiamate anche file di input/output standard. Se si programma per un computer PC compatibile con DOS, sono disponibili due flussi in più, aperti automaticamente quando il programma viene avviato e chiusi autonomamente al termine.

Nome	**Flusso**	**Dispositivo**
stdin	Input standard	Tastiera
stdout	Output standard	Schermo
stderr	Errore standard	Schermo
stdprn *	Stampante standard	Stampante (LPT1:)
stdaux *	Ausiliario standard	Porta seriale (COM1:)

*supportati soltanto in DOS

Ogni volta che si utilizzano le funzioni printf() o puts() per visualizzare del testo sullo schermo si utilizza stdout. Analogamente con la funzione scanf() si utilizza il flusso stdin. I flussi standard sono aperti automaticamente, ma altri, come quelli utilizzati per manipolare i dati memorizzati su disco devono essere aperti esplicitamente.

La libreria standard del C ha molte funzioni per la gestione dei flussi di input e di output. La maggior parte di tali funzioni è disponibile in due varietà: una che utilizza sempre uno dei flussi standard e l'altra che richiede al programmatore di specificare il flusso.

Utilizza uno dei flussi standard	**Richiede un nome di flusso**	**Descrizione**
printf()	fprintf()	Output formattato
vprintf()	vfprintf()	Output formattato con un elenco di argomenti variabile
puts()	fputs()	Output stringa
putchar()	putc(), fputc()	Output carattere
scanf()	fscanf()	Input formattato
gets()	fgets()	Input stringa
getchar()	getc(), fgetc()	Input carattere
perror()		Output stringa solo su stderr

Tutte queste funzioni richiedono l'inclusione del file "stdlib.h". Le funzioni vprintf() e vfprintf() richiedono anche "stdargs.h".

Input da tastiera

La maggior parte dei programmi in C richiede qualche forma di input dalla tastiera (ovvero da stdin). Le funzioni di input si dividono in tre livelli gerarchici: input di carattere, di riga e formattato.

Funzioni per l'input di carattere

Le **funzioni per l'input di carattere** leggono l'input da un flusso un carattere per volta e restituiscono il carattere nel flusso o EOF se è stata raggiunta la fine del file p si è verificato un errore. EOF è una costante simbolica definita in "stdio.h" come –1- Questi funzioni differiscono per buffer e visualizzazione dei caratteri.

- Alcune funzioni dispongono di un buffer dove il sistema operativo memorizza tutti i caratteri fincheè l'utente non preme INVIO per inviarli al flusso stdin. Altre sono prive di buffer, quindi ogni carattere è inviato a stdin appena viene premuto il tasto corrispondente.
- Alcune funzioni di input eseguono l'eco di ogni carattere su stdout non appena viene ricevuto, altri no, perciè il carattere viene inviato a stdin e non a stout.

La funzione getchar()

Questa funzione ottiene il carattere successivo dal flusso stdin. Fornisce un input di carattere bufferizzato con eco e il suo prototipo è:

```
int getchar(void)
```

L'uso di getchar() è mostrato nel listato sotto:

```
#include <stdio.h>

main()
{
      char ch;
      while ((ch = getchar()) != '\n')
            putchar(ch);

      return 0;
}
```

L'output del programma:

```
Questo è ciò che è stato digitato.
Questo è ciò che è stato digitato.
```

La funzione getchar() attende la ricezione di un carattere da stdin. Poichè getchar() è una funzione di input con buffer, non viene ricevuto alcun carattere finchè l'utente non preme INVIO. Tuttavia ogni tasto premuto viene immediatamente mostrato sullo schermo.

Quando si preme INVIO, tutti i caratteri immessi, incluso quello di nuova riga vengono inviati a stdin dal sistema operativo. La funzione getchar() restituisce i caratteri uno alla volta, assegnando ognuno a ch.

Ogni carattere è confrontato con il carattere di nuova riga '\n' e se diverso, visualizzato su schermo con putchar().

Altro esempio dell'utilizzo della funzione getchar() per l'input di un'intera stringa:

```
#include <stdio.h>

#define MAX 80

main()
{
      char ch, buffer[MAX+1];
      int x = 0;

      while (((ch = getchar()) != '\n') && x < MAX)
            buffer[x++] = ch;

      buffer[x] = '\0';     //carattere di fine stringa

      printf("%s\n",buffer);

      return 0;
}
```

Ecco input e output del programma:

```
Questo è ciò che è stato digitato.
Questo è ciò che è stato digitato.
```

Il programma è simile a quello visto in precedenza, per quanto riguarda l'utilizzo di getchar(). Il ciclo presenta una condizione in più: questa volta accetta i caratteri da getchar() finchè viene riconosciuta una nuova riga o dopo 80 caratteri.

L'array buffer ha dimensione MAX+1 e non MAX perché con la dimensione MAX+1 la stringa può contenere 80 caratteri più il terminatore '\0'.

La funzione getch()

Questa funzione ottiene il carattere successivo dal flusso stdin. Fornisce un input di carattere bufferizzato senza eco e il suo prototipo è:

```
int getch(void)
```

Questa funzione non fa parte dello standard ANSI quindi potrebbe non essere disponibile su tutti i sistemi; in più potrebbe richiedere file d'inclusione diversi. In generale il prototipo di getch() è definito in "conio.h". L'uso di getch() è mostrato nel listato sotto:

```
#include <stdio.h>

main()
{
      char ch;

      while ((ch = getch()) != '\n')
            putchar(ch);

      return 0;
}
```

Input e output del programma:

```
Questo è ciò che è stato digitato.
```

Questo è quello che si visualizza in quanto la funzione non ha buffer e getch() restituisce ogni carattere senza attendere la pressione di INVIO. Poiché non viene eseguito l'eco dell'input i caratteri non sono visualizzati sullo schermo.

La funzione getche()

Questa funzione è del tutto simile a getch(), a parte il fatto che esegue l'eco dei caratteri su stdout. Anche questa non è una funzione ANSI, ma molti compilatori la supportano.

Funzioni per l'input di riga

La funzione gets()

Questa funzione si limita a leggere una riga da stdin e la memorizza in una stringa (array di caratteri). LA funzione legge caratteri finchè non incontra un carattere di nuova riga (\n) o la fine della riga.

Prima di utilizzare gets() è necessario allocare uno spazio in memoria sufficiente per la stringa.

```
#include <stdio.h>

main()
{
      char input[81];

      puts("scrivere il testo e premere INVIO:");
      gets(input);
      printf("E\' stato inserito: %s\n",input);

      return 0;
}
```

Ecco input e output del programma:

```
Scrivere il testo e premere INVIO:
Questo è il testo
E' stato inserito: Questo è il testo
```

In questo esempio l'argomento di gets() è l'espressione input, che rappresenta il nome di un array di tipo char.

Funzioni per l'input formattato

La funzione scanf()

Funzione vista nelle lezioni precedenti, utilizzata soprattutto per manipolare i numeri.

Output su schermo

Le funzioni di output sullo schermo si dividono in tre categorie principali come quelle di input: output di caratteri, di riga e di formato.

Output di caratteri

Le funzioni di outpur di caratteri del C inviamo un singolo carattere a un flusso.

La funzione putchar()

Il prototipo della funzione si trova su "stdio.h" ed è il seguente:

```
int putchar(int c);
```

questa funzione scrive il carattere memorizzato nel parametro su stdout. Anche se nel prototipo è specificato un argomento int, in realtà si passa un char. Si può passare un int, purché abbia valore appropriato per un carattere (cioè sia compreso tra 0 e 255). La funzione restituisce il carattere scritto, o EOF in caso di errore.

Output di riga

La funzione puts()

Funzione vista nelle lezioni precedenti.

Output formattato

La funzione printf()

Funzione vista nelle lezioni precedenti.

Funzioni matematiche

La libreria standard del C contiene numerose funzioni che svolgono calcoli matematici. I loro prototipi di trovano nel file di intestazione "math.h". Tutte le funzioni matematiche restituiscono un valore double. Nelle funzioni trigonometriche gli angoli sono espressi in radianti (un radiante equivale a 57,96 gradi; l'angono giro di 360 gradi misura 2π radianti)

Funzioni trigonometriche		
Funzione	**Prototipo**	**Descrizione**
`acos()`	`double acos(double x)`	Restituisce l'arcocoseno dell'argomento. L'argomento deve essere compreso tra –1 e 1. Il valore restituito è compreso tra 0 e π.
`asin()`	`double asin(double x)`	Restituisce l'arcoseno dell'argomento. L'argomento deve essere compreso tra –1 e 1. Il valore restituito è compreso tra -π/2 e π/2.
`atan()`	`double atan(double x)`	Restituisce l'arcotangente dell'argomento. Il valore restituito è compreso tra -π/2 e π/2.
`atan2()`	`double atan2(double x, double y)`	Restituisce l'arcotangente di x/y. Il valore restituito è compreso tra -π e π.
`cos()`	`double cos(double x)`	Restituisce il coseno dell'argomento.
`sin()`	`double sin(double x)`	Restituisce il seno dell'argomento.
`tan()`	`double tan(double x)`	Restituisce la tangente dell'argomento.

Funzioni esponenziali e logaritmiche		
Funzione	**Prototipo**	**Descrizione**
`exp()`	`double exp(double x)`	Restituisce l'esponente naturale dell'argomento cioè e^x (e elevato alla x) dove e è uguale a: 2,7182818284590452354
`log()`	`double log(double x)`	Restituisce il logaritmo naturale dell'argomento. L'argomento deve essere maggiore di zero.
`log10()`	`double log10(double x)`	Restituisce il logaritmo in base 10 dell'argomento. L'argomento deve essere maggiore di zero

`frexp()`	`double frexp(double x, int* y)`	Calcola un numero decimale normalizzato che rappresenta il valore di x. Il valore restituito r è compreso tra 0,5 e 1. La funzione assegna ad y un esponente intero tale che c = r * 2 ^ y. Se l'argomento x vale 0 anche r ed y verranno posti a 0.
`ldexp()`	`double ldexp(double x, int y)`	Restituisce x * 2 ^ y

Funzioni ipergoliche		
Funzione	**Prototipo**	**Descrizione**
`cosh()`	`double cosh(double x)`	Restituisce il coseno iperbolico dell'argomento
`cosh ()`	`double cosh(double x)`	Restituisce il seno iperbolico dell'argomento
`tanh()`	`double tanh(double x)`	Restituisce la tangente iperbolica dell'argomento

Altre funzioni matematiche		
Funzione	**Prototipo**	**Descrizione**
`sqrt()`	`double sqrt(double x)`	Restituisce la radice quadrata dell'argomento che deve essere positivo e non nullo.
`ceil()`	`double ceil(double x)`	Restituisce il più piccolo intero non inferiore all'argomento. Ad esempio ceil(4.5) restituisce 5.0; ceil(-4.5) restituisce -4.0 .
`floor()`	`double floor(double x)`	Restituisce il più grande numero interno non superiore all'argomento. Ad esempio floor(4.5) restituisce 4.0 mentre floor(-4,5) restituisce 5.0.
`abs()`	`int abs(int x)`	Restituisce il valore assoluto dell'argomento.
`labs()`	`long labs(long x)`	Restituisce il valore assoluto dell'argomento.
`modf()`	`double modf(double x, double *y)`	Separa x in una parte intera e una decimale ciascuna con lo stesso segno di x. La parte decimale è il valore restituito; la parte intera è assegnata a *y.

`pow()`	`double pow(double x, double y)`	Restituisce x ^ y (x elevato alla y); si genera errore se x = 0 e y <= 0 o se x < 0 e y non è intero.
`fmod()`	`double fmod(double x, double y)`	Restituisce il resto in formato decimale di x/y. Con lo stesso segno di x. Restituisce 0 se x è uguale a 0.