I/O da tastiera e direttive processore

Alessandra Giordani
agiordani@disi.unitn.it
Lunedì 3 maggio 2010
http://disi.unitn.it/~agiordani/

Ripasso funzione printf()

- Usata per stampare il contenuto di una o più variabile
- All'interno della stringa **template** possiamo inserire delle sequenze di caratteri speciali a cui, in fase di stampa, l'interprete sostituirà il valore attuale di una variabile.
- Nel **template** dobbiamo 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
  - per stampare una variabile stringa, la sequenza speciale è %s

  (stringa terminata da *null*)

printf() - Esempio

```
char a = 'B';
int x = 1000, y = 20000;
float k = 50.5;
printf("la variabile a vale: %c", a);
printf("la variabile x vale: %d", x);
printf("la variabile y vale: %d", y);
printf("la variabile k vale: %f", k);
printf("le variabili: k,x,a valgono rispettivamente:
                %f,%d,%c", k,x,a );
```

- Stampa “la variabile k vale: 50.500000”
- Se specificiamo “%.3f” stampa 50.500
- La precisione (es. 3) indica quante cifre devono essere stampate dopo il .

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() - Esempio

- Chiedi all'utente di immettere un numero e calcolane la radice:

```c
#include <stdio.h>
#include <math.h>
int main() {
    double number;
    printf("Inserisci un numero decimale: ");
    scanf("%lf", &number);
    printf("La radice quadrata di %f é %f\n",
       number, sqrt(number));
    return 0;
}
```

- All'interno del template:

`%d` Legge un intero (si aspetta un `int*`)

`%f` Legge un float (si aspetta un `float*`)

`%lf` Legge un double (si aspetta un `double*`)

`%s` Legge una stringa (si aspetta un `char*`, vedi slide successive)

- Uno spazio consuma qualunque sequenza di white-space characters (spazi, tabulazioni, a capo)

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

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

Input di stringhe

- Queste considerazioni sono particolarmente delicate nel caso in cui vogliamo usare scanf per leggere stringhe
- La sequenza %s all'interno di un template consuma qualunque sequenza di caratteri che non contenga spazi
- Per leggere una stringa, dobbiamo prima avere allocato un buffer (cioe un array di caratteri) abbastanza capiente. Ad esempio:

```
char buf [256]; // crea un buffer
scanf("%s", buf);
```

- Se la stringa immessa dall'utente è piu lunga del buffer (nell'esempio la lunghezza massima è 255, tenendo conto del null character) abbiamo un errore di **segmentation fault** (violazione di accesso in memoria)

scanf() - matching multiplo

- La funzione scanf puo' essere usata per leggere più di una variabile in un colpo solo:

```
#include <stdio.h>
int main()
{
    float peso, altezza;
    printf("Immetti il tuo peso in kg e la tua altezza in metri:\n");
    scanf("%f %f",&peso,&altezza);
    float imc=peso/(altezza*altezza);    //indice di massa corporea
    printf("Il tuo IMC e' %f. ",imc);
    if (imc<18.5)
       printf("Sei sottopeso, mangia!\n");
    else if(imc<25)
         printf("Alla grande!\n");
    else if (imc<30)
         printf("Sei un po' in carne, datti una regolata!\n");
    else printf("A dieta, subito!\n");
    return 0;
}
```

Il preprocessore

- Il preprocessore C è un programma che viene eseguito prima del compilatore (non è necessario "lanciarlo" esplicitamente)
- Attraverso il preprocessore si esprimono direttive al compilatore
- Il preprocessore ha la sua grammatica e la sua sintassi che sono scorrelate da quelle del C
- Ogni direttiva inizia con il simbolo #, che deve essere il primo carattere diverso dallo spazio sulla linea
- Le direttive del preprocessore terminano con un newline (non con ";")

Direttive del preprocessore

- Principali compiti richiesti al preprocessore:
  - Inclusione del codice sorgente scritto su altro file
  - Definizione delle costanti simboliche
  - Compilazione condizionale del codice

La direttiva #include – 1

- La direttiva #include fa sì che il compilatore legga il testo sorgente da un file diverso da quello che sta compilando
- #include lascia inalterato il file da cui vengono prelevati i contenuti
  - Utile quando le stesse informazioni devono essere condivise da più file sorgente: si raccolgono le informazioni comuni in un unico file e lo si include ovunque sia necessario
  - Si riduce la quantità di testo da digitare e si facilita la manutenzione: i cambiamenti al codice condiviso hanno effetto immediato su tutti i programmi che lo includono

La direttiva #include – 2

- La direttiva #include può assumere due formati

```
#include <nome_file.h>

#include "nome_file.h"
```

  - …nel primo caso, il preprocessore cerca il file in una directory speciale, definita dall'implementazione del compilatore, dove sono contenuti i file che vengono normalmente inclusi da tutti i programmi utente (sintassi usata per includere file di intestazione, *header file*, della libreria standard)
  - …nel secondo caso, il file viene prima cercato nella directory del file sorgente e, quando non reperito, seguendo il percorso classico

La direttiva #define

- La direttiva #define consente di associare un nome ad una costante
- Esempio:

```
#define  NIENTE  0
```

  associa il nome "NIENTE" alla costante 0
- Per evitare confusione fra nomi di costanti e nomi di variabili, è pratica comune usare solo lettere maiuscole per le costanti e solo minuscole per le variabili
- L'associazione di nomi alle costanti permette…
  - …di utilizzare un nome descrittivo per oggetti altrimenti non autoreferenziali
  - …di semplificare la modifica del software: cambiare il valore ad una costante equivale a cambiarne la sola definizione e non tutte le occorrenze

La compilazione condizionale

- Consente al programmatore di controllare la compilazione del codice del programma
- Esempio:

```
#define DEBUG 1
#ifdef DEBUG
      printf("Variabile x = %d\n",x);
#endif
```

l'istruzione *printf()* viene compilata (ed eseguita) solo nel caso in cui la costante DEBUG sia definita

Esempio sulle direttive

```c
#include <stdio.h>
#define MAX 5

int main()
{
    int a[MAX],b[MAX],c[MAX];
    int i,tmp;

    for (i=0;i<MAX;i++)
    {
        printf("Inserire il %d° numero del vettore a: ", i+1);
        scanf("%d",&tmp);
        a[i]=tmp;
    }
    for (i=0;i<MAX;i++)
    {
        printf("Inserire il %d° numero del vettore b: ", i+1);
        scanf("%d",&tmp);
        b[i]=tmp;
    }
    printf("\nContenuto del vettore c:\n", i+1);
    for (i=0;i<MAX;i++)
    {
        c[i]=a[i]+b[i];
        printf("[%d]",c[i]);
    }
    printf("\n");
    return 0;
}
```

Esempio sulle direttive

```c
#include <stdio.h>
#define MAX 5
```

```c
#define RIEPILOGO 1
```

```c
int main()
{
    int a[MAX],b[MAX],c[MAX];
    int i,tmp;

    for (i=0;i<MAX;i++)
    {
        printf("Inserire il %d° numero del vettore a: ", i+1);
        scanf("%d",&tmp);
        a[i]=tmp;
    }
    for (i=0;i<MAX;i++)
    {
        printf("Inserire il %d° numero del vettore b: ", i+1);
        scanf("%d",&tmp);
        b[i]=tmp;
    }
    printf("\nContenuto del vettore c:\n", i+1);
    for (i=0;i<MAX;i++)
    {
        c[i]=a[i]+b[i];
        printf("[%d]",c[i]);
    }
    printf("\n");
    return 0;
}
```

```c
#ifdef RIEPILOGO
printf("\nContenuto del vettore a:\n", i+1);
for (i=0;i<MAX;i++)
    printf("[%d]",a[i]);
printf("\nContenuto del vettore b:\n", i+1);
for (i=0;i<MAX;i++)
    printf("[%d]",b[i]);
#endif RIEPILOGO
```

Differenza 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;
}
```