Corso di laurea in fisica
Anno Accademico 2020/2021
Laboratorio di informatica 1

Esercitazione 4

Nota 1 I caratteri alfanumerici e alcuni segni di punteggiatura e speciali sono rappresentati, nella memoria di un calcolatore, da numeri interi, sulla base di un codice convenzionale. In C, il tipo `char` comprende quindi valori che sono di fatto numeri interi, e sono quindi in parte compatibili con valori di tipo `int`. Le operazioni fondamentali sui caratteri sono le seguenti. (1) Assegnamento: `c = 'm'` assegna alla variabile `c` il valore numerico della lettera 'm' minuscola; (2) confronto: `(c == 'm')` è un'espressione, vera se la variabile `c` contiene il valore numerico corrispondente alla lettera 'm' minuscola. Attenzione: il valore numerico della cifra '7' non è il numero 7!

Le funzioni `getchar` e `putchar` permettono, rispettivamente, di leggere un carattere battuto alla tastiera e di stampare sul terminale un carattere. I caratteri battuti alla tastiera durante l'esecuzione di un programma vengono di norma accumulati in un *buffer* (che possiamo considerare come un'area di parcheggio temporanea) dal sistema operativo, che li "consegna" al programma solo quando si digita il carattere di "a capo" (`RETURN` o `INVIO`). Ogni chiamata della funzione `getchar` consuma uno di questi caratteri. L'istruzione `c = getchar()` assegna quel carattere (o meglio, la sua codifica numerica) alla variabile `c`. La funzione `putchar` stampa sul terminale il carattere che le viene passato come argomento. La sigla `EOF` rappresenta un carattere speciale, che serve a segnalare la fine del flusso di caratteri in ingresso al programma. Nei sistemi operativi della famiglia *Unix*, questo carattere speciale viene prodotto quando si digita la combinazione di tasti `CTRL-D`, cioè quando si premono contemporaneamente i tasti `CTRL` e `d`.

Nota 2 La dimensione dello spazio occupato da una variabile nella memoria di lavoro dipende dal tipo della variabile. Il linguaggio C, però, non prescrive valori specifici, che possono quindi variare da un calcolatore a un altro. Per conoscere le dimensioni associate ai tipi in un caso particolare, si può usare, in un programma, la funzione `sizeof`, che richiede come argomento il nome di un tipo o l'identificatore di una variabile. La funzione `sizeof` restituisce un valore di tipo `size_t`, che può equivalere a `unsigned int` o a `unsigned long int`; la sigla da utilizzare nella chiamata della funzione `printf` è `%zu`.

1 Scrivete un programma che, servendosi delle funzioni `getchar` e `putchar`, legge una sequenza di caratteri dallo *standard input*, arrestandosi quando il carattere è l'asterisco (`*`), e stampa la stessa sequenza sostituendo ogni 'a' con una 'e'. Il programma non deve trattenere in memoria la sequenza (modificate il programma `filtro.c`).

2 Definite un tipo (*record*) adeguato a rappresentare i punti del piano cartesiano; la rappresentazione deve comprendere anche una lettera, che interpretiamo come nome simbolico di un punto.

Scrivete una funzione che riceve due punti come argomenti e calcola la loro distanza. Scrivete una funzione che riceve un punto come argomento e calcola la norma del vettore corrispondente.

Definite un tipo (*record*) per rappresentare circonferenze nel piano cartesiano. Quali informazioni sono sufficienti per determinare univocamente una circonferenza? Scrivete una funzione che riceve, come argomenti, un punto e una circonferenza, e stabilisce se il punto è interno alla circonferenza, esterno, oppure sulla circonferenza. Scrivete una funzione che riceve, come argomenti, due circonferenze, e stabilisce se la seconda è interna alla prima.

3 Definite un tipo di dati adeguato a rappresentare un rettangolo nel piano cartesiano, con i lati paralleli agli assi (ragionate su quali informazioni siano sufficienti a determinare univocamente il rettangolo).

Scrivete una funzione che, dato un rettangolo, ne calcola l'area.

Scrivete una funzione che, dati due rettangoli, calcola l'area della regione in cui si sovrappongono (se i due rettangoli non si sovrappongono, o si sovrappongono solo in un punto o in un lato, la funzione deve restituire il valore 0).

Incorporate le due funzioni in un programma che collauda le due funzioni in varie circostanze.

4 Questo esercizio deve essere svolto in gruppo e consegnato entro il 22 aprile all'indirizzo `luca.bernardinello@unimib.it`. *L'oggetto del messaggio deve consistere nella sigla LABINF, seguita dal numero di matricola di uno degli studenti del gruppo. Il file con il codice sorgente della soluzione deve avere come nome lo stesso numero di matricola indicato nell'oggetto del messaggio, seguito dall'estensione* `.c`. *La prima riga del file deve consistere in un commento che riporta i numeri di matricola dei tre componenti del gruppo, separati da spazi, come in questo esempio:*

```
// 123456 789012 345678
```

Consideriamo un vettore A, di lunghezza N, e un vettore B, di lunghezza M, contenenti numeri interi in ordine crescente. Ad esempio, consideriamo $A = (-6, -2, 9, 13, 21)$ e $B = (-1, 0, 12)$. Vogliamo "fondere" A e B in un terzo vettore, C, di lunghezza $N + M$, in modo che anche C sia ordinato: $C = (-6, -2, -1, 0, 9, 12, 13, 21)$.

Dovete scrivere un programma che legge i valori di A e di B dallo *standard input*, li memorizza in due vettori, e compie l'operazione di fusione. Questa

operazione dev'essere compiuta in una funzione. Il programma deve stampare il contenuto finale del terzo vettore. Supponete che i dati siano presentati in due righe successive, come nell'esempio qui sotto:

```
-6 -2 9 13 21
-1 0 12
```

Il programma dev'essere scritto supponendo che i dati siano presentati correttamente, cioè che le due righe contengano rispettivamente N e M numeri interi in ordine crescente. I valori di M e N devono essere fissati in direttive `define`. La funzione che compie la fusione deve operare correttamente per qualsiasi lunghezza dei due vettori, sfruttando eventualmente gli argomenti.