Laboratorio di programmazione

18 gennaio 2008

Molti degli esercizi proposti oggi prevedono di modificare alcuni programmi preparati nelle scorse lezioni utilizzando i puntatori in modo avanzato, ad esempio per migliorare l'occupazione della memoria.

Figure geometriche (vedi 11 gennaio 2008)

Aggiungete al vostro programma una funzione che crei una nuova figura e la inizializzi con dati inseriti dall'utente, allocando la memoria necessaria con l'uso della funzione `malloc`.

Rovescia (vedi 14 dicembre 2007)

Scrivete un programma che legga una sequenza di numeri interi terminata da 0 e li stampi dall'ultimo (0 escluso) al primo. Non potendo prevedere quanti numeri verranno inseriti, allocate la memoria necessaria in modo dinamico (ad esempio raddoppiando la lunghezza del vettore ogni volta che questo si riempie).

Rubrica (vedi 11 gennaio 2008)

Modificate il programma per la gestione della rubrica in modo da non occupare spazio inutile, allocando dinamicamente la memoria necessaria a contenere le stringhe inserite.

Create poi una seconda versione del programma che memorizzi le voci della rubrica in una lista anzichè in un array di strutture.

Funzione somma

Considerate il seguente prototipo di funzione: `int somma( int (*f) (int), int inizio, int fine)`. Qual è il nome della funzione? Quanti parametri ha? Di che tipo sono questi parametri? Che tipo restituisce?

Scrivete la suddetta funzione, sapendo che la chiamata `somma(g, i, j)` deve restituire il valore `g(i) + ⋯ + g(j)`.

Manipolazione di un array di interi

Usate typedef per definire un tipo di funzione chiamato `Fint2int` che ha un parametro intero e restituisce un intero. Definite poi tre funzioni di questo tipo chiamate `quadrato`, `triplo`, `opposto` che trasformino l'intero passato come argomento (ad es: 4) nel suo quedrato (es: 16), nel suo triplo (es: 12), nel suo opposto (es: -4) rispettivamente.

Quindi scrivete una funzione con prototipo `void apply ( Fint2int *f, int a[], int n )` che applichi alle prime n componenti del vettore di interi `a[]` la funzione puntata da `f`.

Ordinamento di un vettore

La funzione `qsort` permette di ordinare un vettore (o una porzione di vettore) di elementi di tipo generico; `qsort` ha il seguente prototipo:

```
void qsort ( void * base, size_t nmemb, size_t size,
int (*compar) (const void *, const void * )
```

dove `base` punta al primo elemento del vettore, `nmemb` è il numero di elementi da ordinare, `size` è la dimensione di ogni elemento, `compar` è un puntatore alla funzione da usare per confrontare due elementi. La funzione `compar` va definita sulla base del tipo di oggetti che si vogliono ordinare.

Scrivete due programmi che ordinano rispettivamente un vettore di interi e un vettore di stringhe, usando `qsort`. Per farlo, dovrete definire due funzioni `compar_int` e `compar_string` in grado di confrontare due interi e due stringe, rispettivamente. Quali prototipi dovranno avere tali funzioni per essere usate come argomento di `qsort`?

Menu di comandi per trasformare stringhe

Inventate e scrivete alcune funzioni che trasformino una stringa passata come argomento (es: tutto maiuscolo, tutto minuscolo, lettere spaziate, stringa rovesciata.... ovvero, se se la stringa di input fosse `strINGA`, si potrebbero definire funzioni che la trasformano in `STRINGA`, `stringa`, `s t r I N G A`...)

Scrivete quindi un programma che proponga un menu di possibili trasformazioni, chieda all'utente di sceglierne una e la applichi, gestendo il menu non con uno switch, ma con un vettore di puntatori alle funzioni applicabili alla stringa.