Laboratorio di Programmazione: Linguaggio C
Lezione 16 del 17 marzo 2014

Damiano Macedonio

Esercizio 1

Scrivere un programma C che analizza la relazione di inclusione tra rettangoli nel piano cartesiano (di coordinate reali). Il programma chiede prima 10 rettangoli, poi un rettangolo `r` e conta quanti dei 10 rettangoli inseriti sono uguali a `r` e quanti contengono `r`.

La struttura *punto* è composta da due `double` che rappresentano l'ascissa e l'ordinata dei punti sul piano cartesiano.

La struttura *rettangolo* rappresenta i rettangoli con i lati paralleli agli assi cartesiani. La rappresentazione di tale struttura è normalizzata. Essa è composta da due punti che rappresentano due vertici opposti del rettangolo: `max` (il vertice con ascissa e ordinata massime) e `min` (il vertice con ascissa e ordinata minime).

Definizione: Diciamo che il rettangolo `r1` è *incluso* nel rettangolo `r2` se i lati di `r1` sono contenuti nella parte finita di piano delimitata dal perimetro di `r2`, perimetro incluso.

Il programma deve includere le seguenti funzioni:

`struct rettangolo crea_rettangolo(struct punto a, struct punto b)` riceve due punti `a`, `b` e restituisce il rettangolo che ha `a` e `b` come vertici opposti.
Attenzione: Non si deve richiedere che i due vertici siano quelli della rappresentazione normalizzata dei rettangoli.

`bool include(struct rettangolo this, struct rettangolo that)` verifica se il rettangolo `this` include il rettangolo `that`.

`bool equals_rettangoli(struct rettangolo this, struct rettangolo that)` verifica se i rettangoli `this` e `that` sono uguali.

`int main(void)` Chiede all'utente 10 rettangoli e li memorizza in un array `a`. Chiede all'utente un rettangolo `r`. Scorre `a` per verificare quanti rettangoli sono uguali a `r` e quanti contengono `r`.
Attenzione: Per definire i rettangoli, l'utente deve inserire due vertici opposti. Tali vertici non sono necessariamente quelli usati nella rappresentazione normalizzata dei rettangoli.

`...` opportune funzioni ausiliarie.

Esercizio 2

Scrivere un programma C per la registrazione e lo studio delle condizioni meteorologiche in un certo periodo.

Le condizioni meteorologiche di un giorno sono raccolte in una struttura *dati meteo* composta da: una struttura per la registrazione di *data* e *ora*, un `float` contenente la *temperatura*, un `float` contenente la *pressione atmosferica*, un `float` contenente la *percentuale di umidità* e un `float` contenente la *quantità di pioggia* caduta.

Assumete che il programma gestisca al massimo 30 registrazioni in una struttura archivio formata da un array di *dati meteo* e un intero rappresentante il numero di dati registrati.

Il programma è composto dalle seguenti funzioni:

`void stampa_menu(void)` stampa a video la lista delle operazioni che possono essere eseguite dall'utente: registra nuovo dato meteorologico, calcola statistiche, esci dal programma.

`char leggi_scelta(void)` richiede all'utente quale tra le possibili operazioni vuole eseguire e restituisce un carattere rappresentante tale scelta.
Attenzione: Ricordatevi di usare la stringa di formato `" %c"` con uno spazio davanti per eliminare qualsiasi carattere di spazio/invio precedentemente inserito.

`struct archivio registra(struct archivio a, struct dati_meteo d)` registra un nuovo insieme di dati meteo `d` nell'archivio `a` e restituisce l'archivio aggiornato.
Attenzione: Ricordatevi che le modifiche effettuate dalla funzione sull'archivio non sono visibili all'esterno, quindi l'archivio aggiornato va restituito come risultato della funzione e assegnato alla variabile precedente.

`struct statistiche calcola(struct archivio a)` ritorna una struttura contenente le statistiche dei dati inseriti. La struttura `struct statistiche` contiene i valori minimi/massimi/medi di temperatura/pressione/umidità/pioggia .

`int main(void)` richiama le funzioni `stampa_menu()` e `leggi_scelta()`, ed esegue l'operazione richiesta, finché la scelta effettuata dall'utente non è quella di uscire.

Esercizio 3

Scrivere un programma C che gestisce un libretto universitario.

- Un *libretto universitario* è una struttura formata da: un array di 20 esami ed un intero rappresentante il numero di esami registrati.
- Un *esame* è una struttura formata da: una data di registrazione, un codice di tre caratteri rappresentante il nome del corso, un codice di due caratteri rappresentante le iniziali del docente, un intero rappresentante il voto, un booleano per la lode.
- Una *data* è una struttura formata da 3 interi: giorno, mese, e anno.

Il programma è composto dalle seguenti funzioni:

`void stampa_menu(void)` stampa a video la lista delle operazioni che possono essere eseguite dall'utente: registra esame, stampa esami, calcola media, conta lodi, esci dal programma.

`char leggi_scelta(void)` richiede all'utente quale tra le possibili operazioni vuole eseguire e restituisce un carattere rappresentante tale scelta.
Attenzione: Ricordatevi di usare la stringa di formato `" %c"` con uno spazio davanti per eliminare qualsiasi carattere di spazio/invio precedentemente inserito.

`void stampa_esami(struct libretto l)` stampa gli esami contenuti nel libretto `l`.

`int media(struct libretto l)` calcola la media dei voti registrati nel libretto `l`.

`int num_lodi(struct libretto l)` restituisce il numero di lodi registrate nel libretto `l`.

`struct libretto registra_esame(struct libretto l, struct esame e)` registra l'esame `e` nel libretto `l` e restituisce il libretto aggiornato.
Attenzione: Ricordatevi che le modifiche effettuate dalla funzione sul libretto non sono visibili all'esterno, quindi il libretto aggiornato va restituito come risultato della funzione e assegnato alla variabile precedente.

`int main(void)` richiama le funzioni `stampa_menu()` e `leggi_scelta()`, ed esegue l'operazione richiesta, finché la scelta effettuata dall'utente non è quella di uscire.