

Progetto – "Agenda elettronica"

Realizzare un programma C++ che offra funzioni di base per la gestione di una semplice agenda elettronica personale, per uno specifico anno. Precisamente, il programma dovrà realizzare un tipo di dato **agenda** che contenga le seguenti *informazioni*:

- **dati personali** del proprietario dell'agenda (vedi sotto)
- elenco completo dei **giorni** dell'anno con informazioni per ogni giorno sugli **eventi** del giorno (vedi sotto), ed eventuali **note** generali (= testo libero, max 200 caratteri)
- **rubrica telefonica** (= sequenza di informazioni del tipo <nome, cognome, indirizzo, numeri di telefono, email>).

e permetta l'esecuzione delle seguenti *operazioni*:

- immissione dei dati personali del proprietario
- immissione di un evento per uno specifico giorno
- immissione di note generali per uno specifico giorno
- immissione di una nuova persona (tutti i suoi dati) nella rubrica telefonica
- visualizzazione dei dati personali e modifica di indirizzo e telefono
- ricerca e visualizzazione di tutti gli eventi e delle note di uno specifico giorno
- ricerca e visualizzazione di tutti i giorni in cui è presente uno specifico evento
- ricerca e visualizzazione di tutti i numeri di telefono di una specifica persona nella rubrica telefonica
- cancellazione di uno specifico evento o delle note per uno specifico giorno
- cancellazione di una specifica persona dalla rubrica telefonica
- modifica dei dati (indirizzo, numeri di telefono, email) di una specifica persona nella rubrica telefonica
- salvataggio su file di tutti i dati
- stampa (su std output) del calendario di uno specifico mese dell'anno [FACOLTATIVO].

N.B. Uno specifico giorno è individuato tramite la sua data completa; uno specifico evento è individuato tramite una **sottostringa** della descrizione dell'evento; una specifica persona è individuata tramite il suo cognome + nome.

I **dati personali** comprendono:

- nome e cognome
- indirizzo, numero telefono, e-mail
- data di nascita
- codice fiscale

La **data del giorno** comprende:

- giorno, mese, anno
- giorno della settimana
- numero progressivo del giorno dall'inizio dell'anno

Un **evento** può essere un impegno o una ricorrenza. Un **impegno** comprende:

- orario d'inizio dell'impegno
- descrizione (testo libero, max 100 caratteri; ad es.: incontro, riunione, esame, lezione, ...)
- luogo
- eventuale nome e cognome della persona da incontrare, ...

Una **ricorrenza** comprende:

- descrizione (testo libero, max 200 caratteri; ad es.: compleanno, anniversario, scadenza, viaggio, ...).

Ogni giorno comprende una serie di impegni (max. 10) e di ricorrenze (max. 5).

Interazione con l'utente

- L'interazione avviene tramite **menù a più livelli**. Il programma provvede anche ad effettuare controlli sulla **correttezza dei dati** immessi (ad esempio, codice dell'operazione scelta, validità delle date, ecc.).
- Quando il programma viene attivato richiede l'**anno** a cui ci si riferisce. Se è la prima volta che si opera sull'agenda per quell'anno, il programma richiede tutti i dati personali. N.B. E' possibile creare agende dal 2010 in avanti.

Note di implementazione

- Le elaborazioni (immissione, ricerche, modifiche) avvengono soltanto su dati memorizzati **in memoria principale**. I dati relativi a dati personali, eventi dell'anno e rubrica telefonica vengono salvati su (uno o più) **file** su richiesta esplicita dell'utente e/o automaticamente al termine del programma, e caricati da file in memoria principale all'inizio dell'esecuzione del programma. I nomi dei file contengono indicazione dell'anno cui ci si riferisce (ad es., 'eventi2010.txt'). Se i file relativi all'anno scelto non esistono vengono creati alla prima operazione di salvataggio dati.
- L'intera **agenda**, il singolo **giorno**, il singolo **impegno**, i **dati personali** e la **data del giorno** devono essere realizzati come altrettante **classi** (obbligatorio). Le classi dati personali e data del giorno estendono (tramite ereditarietà), rispettivamente, le due classi base **persona** e **data** mostrate a lezione (si veda <http://www.math.unipr.it/~gianfr/Teaching/FondProgr/Esempi>) (obbligatorio).
- Si richiede di realizzare la gestione di eventuali situazioni anomale tramite i meccanismi di **gestione eccezioni** del C++ ove opportuno.
- E' vietato: usare dichiarazioni `friend`; definire classi con solo parte `public`; usare ereditarietà a sproposito, usare librerie grafiche.