Corso di Algoritmi e Strutture Dati
Esercitazione 3: alberi e visite

Carlo Linneo (1707 – 1778), medico e naturalista svedese, propose una tra le prime tassonomie degli organismi viventi i cui principi fondamentali sono ancora oggi riconosciuti validi. La versione moderna di tale tassonomia si struttura su un insieme di livelli che, dal più generico al più specifico, si chiamano: *dominio*, *regno*, *phylum*, *classe*, *ordine*, *famiglia*, *genere*, *specie*. Oltre a questi, sono anche previsti alcuni sottolivelli. In tale sistema la classificazione delle specie umana, *Homo sapiens sapiens*, è la seguente:

```
Dominio          Eukaryota
Regno            Animalia
Phylum           Chordata
Subphylum        Vertebrata
Classe           Mammalia
Sottoclasse      Eutheria
Ordine           Primates
Sottordine       Haplorrhini
Superfamiglia    Hominoidea
Famiglia         Hominidae
Genere           Homo
Specie           H. sapiens
Sottospecie      H. s. sapiens
```

ed è rappresentata graficamente in Figura 1, dove compaiono anche nostri "cugini" e "zii". Si noti che l'albero tassonomico *non è un albero genealogico*.

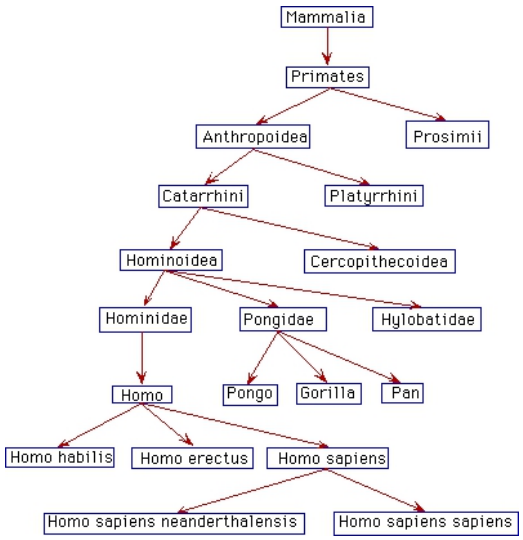

Figure 1: Classificazione di *Homo sapiens sapiens* e di alcuni suoi "parenti".

Scopo di questa esercitazione è implementare le strutture dati per gestire adeguatamente una tassonomia di esseri viventi.

Nel dominio considerato, ogni nodo dell'albero **può avere un numero arbitrario di figli** ed il nome scientifico che lo etichetta è **unico** nell'albero e può quindi identificare univocamente il nodo stesso. Tale nome scientifico non è altro che una stringa di caratteri

Si implementi il tipo di dato `Albero` di elementi di tipo stringa, in cui gli elementi si suppongono unici (nel senso che non sono ammessi più nodi con la stessa stringa) e in cui siano possibili le operazioni seguenti:

- Creazione di un albero vuoto.
- Controllo per stabilire se un albero è vuoto.
- "Affiliazione": inserimento di un nuovo nodo nell'albero, indicando, come parametri dell'operazione, la stringa che identifica il padre e la stringa che rappresenta il nuovo figlio. Se il padre non appartiene all'albero, oppure se esiste già un altro nodo con la stessa stringa del nuovo figlio, la funzione deve restituire `false` e il nuovo nodo non deve essere aggiunto all'albero.
- Visita dell'albero, con una strategia di visita a scelta tra quelle conosciute **implementata in maniera iterativa e non ricorsiva**. La visita ha lo scopo di visualizzare l'intero contenuto dell'albero.

Si noti che la funzione di affiliazione richiede in generale la ricerca di un particolare nodo all'interno dell'albero, e ciò richiede di eseguire una strategia di visita, che però si interrompe se e quando il nodo viene trovato (a differenza della visita pura e semplice, che termina dopo aver visitato tutto).

Il tipo di dato `Albero` deve essere implementato come *struct* C++, e le operazioni sopra richieste devono essere realizzate come metodi di tale *struct*, rispettando il **principio di incapsulamento**.

Nota bene: In questa esercitazione si richiede che `Albero` sia implementato mediante strutture dati collegate. **Non è consentito usare array né** `vector` né classi C++ diverse da quelle per la gestione dell'IO e delle stringhe.

La struct C++ implementata deve essere impiegate in un semplice programma C++, che deve svolgere le azioni seguenti:

1. Creazione dell'albero acquisendo da standard input le informazioni che caratterizzano i nodi (nel nostro caso le stringhe di caratteri). Il formato dell'input deve essere il seguente:

   ```
   padre1 figlio1.1 figlio1.2 figlio1.3 .... figlio1.N
   padre2 figlio2.1 figlio2.2 ..............figlio 2.M
   ....
   padreK figlioK.1 figlioK.2 ..............figlio K.H
   0
   ```

   dove il padre inserito per primo deve necessariamente identificare la radice e i padri successivi devono essere stati precedentemente inseriti nell'albero come figli di qualche altro nodo. Il programma deve rilevare l'eventuale presenza di padri non precedentemente inseriti e in tal caso visualizzare un messaggio di errore (e terminare).
   Come esempio di input corretto, ecco una possibile costruzione dell'albero di Figura 1:

   ```
   mammalia primates
   primates anthropoidea prosimii
   ```

```
anthropoidea catarrhini platyrrhini
catarrhini hominoidea cercopithecoidea
hominoidea hominidae pongidae hylobatidae
hominidae homo
homo homoAbilis homoErectus homoSapiens
homoSapiens homoSapiensNeanderthaliensis homoSapiensSapiens
pongidae pongo gorilla pan
0
```

2. Visualizzazione dell'albero creato, nel senso di mostrare l'intero contenuto dell'albero.