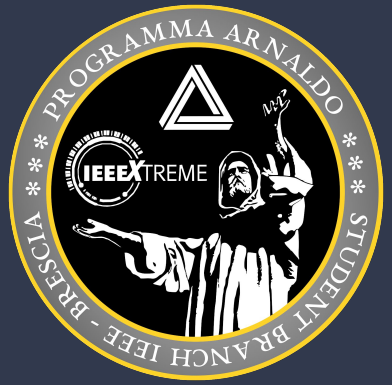

PROGRAMMA ARNALDO

Codice Fiscale

A.A 2022/23

Codice Fiscale

Problema: dato un file XML contenente una lista di dati di persone, generarne i codici fiscali e verificarne la presenza in un secondo file XML, di cui si verifichi la validità.
Scrivere i risultati in un ultimo file XML di output.

Esempio 1

I dati forniti sono:
Nome: Ilaria (→ LRI)
Cognome: Pasini (→ PSN)
Sesso: F
Data di nascita: 1999-04-24 (→ 99D64)
Luogo di nascita: Clusone (→ C800)
Carattere di controllo: (→ M)

Codice Fiscale: PSNLRI99D64C800M

Per ulteriori informazioni sulla generazione dei codici fiscali, vedasi:

Codice fiscale - Wikipedia

Nota:
Nella generazione del codice, può essere tranquillamente trascurata la parte riguardante l'omocodia che compare nella pagina di Wikipedia

Formato dei file di input *[inputPersone.xml]*

Il primo file, **inputPersone.xml**, contiene i dati anagrafici di un certo numero di persone (specificato come attributo numero del tag persone)

Modello:

```
<persone numero=”20”>
  <persona id=”0”>
    <nome>Ilaria</nome>
    <cognome>Pasini</cognome>
    <sesso>F</sesso>
    <comune_nascita>Clusone</comune_nascita>
    <data_nascita>1999-04-24</data_nascita>
  </persona>
  ...
</persone>
```

Formato dei file di input *[comuni.xml]*

Il secondo file, **comuni.xml**, contiene il codice identificativo di ogni comune, da utilizzare nella generazione delle lettere 12-15 del codice fiscale (il numero è l'attributo *numero* del tag comuni)

Modello:

```
<comuni numero="20">
  <comune>
    <nome>ABANO TERME</nome>
    <codice>A001</codice>
  </comune>
  <comune>
    <nome>ABBADIA CERRETO</nome>
    <codice>A002</codice>
  </comune>
  ...
</persone>
```

Formato dei file di input *[codiciFiscali.xml]*

Il terzo file, **codiciFiscali.xml**, contiene un certo numero di codici fiscali (specificato come attributo *numero* del tag codici)

Modello:

```
<codici numero=”9”>
  <codice>PSNLRI99D64C800M</codice>
  <codice>BLDVNI90T45D585A</codice>
  <codice>CNAMRN15H50F751D</codice>
  <codice>STLSPR35T49B196T</codice>
  <codice>GRDVTR83A42F987S</codice>
  <codice>CMPDNI86T68D554P</codice>
  <codice>SGERLA89P51D108C</codice>
  <codice>VNTLGN28C12B490Q</codice>
  <codice>SMNLJN06S62L986S</codice>
</persone>
```

Traccia dell'esercizio

Passo 1:
Leggere tutti i dati di tutte le persone dal file **inputPersone.xml**

Passo 2:
Generare i codici fiscali di tutte le persone, appoggiandosi al file **comuni.xml** per quanto riguarda il codice del comune

Passo 3:
Verificare la validità dei codici fiscali nel file **codiciFiscali.xml**.
Verificare se il codice fiscale di ogni persona risulta presente nel file codiciFiscali.xml. Il risultato di questo passo influenzerà la scrittura del file di output.

Passo 4:
Scrivere un file XML (**codiciPersone.xml**), contenente le informazioni necessarie (presentate nella prossima slide)

Formato dei file di output *[codiciPersone.xml]*

Il file di output, **codiciPersone.xml**, andrà generato nella cartella del progetto e dovrà contenere i seguenti dati:

- un elemento radice chiamato <output>
- al suo interno, due elementi, <persone> e <codici>

Persone: sarà necessario inserire un attributo *numero* che specifichi il numero di persone
All'interno di questo elemento si dovranno trovare i dati anagrafici di tutte le persone presenti in *inputPersone.xml* e un elemento <codice_fiscale> contenente:

- il codice fiscale della persona, se esso è presente in *codiciFiscali.xml*
- la dicitura **ASSENTE** altrimenti

Codici:
Questa sezione contiene due elementi, <invalidi> e <spaiati>, ciascuno con un attributo *numero*.
Nel primo andranno si troveranno i codici fiscali invalidi, nel secondo ci saranno i codici fiscali che non corrispondono a nessuna persona

Nelle prossime slide è meglio spiegata la questa struttura con un esempio

Controlli necessari sui Codici Fiscali

Per la verifica di validità del codice fiscale sono previsti i seguenti controlli:

- Controllo caratteri e cifre nelle posizioni corrette
  [PSNLRI99D64C8MG0]]
- Validità del giorno (compreso fra 1 e 31 o fra 41 e 71)
  [PSNLRI99D94C800M]
- Validità del mese (valori ammessi: A, B, C, D, E, H, L, M, P, R, S, T)
  [PSNLRI99K64C800M]
- Correttezza carattere di controllo (stesso algoritmo usato nella generazione del codice)
  [PSNLRI99D64C800K]
- Numero di giorni in un mese (per evitare eccessive complicazioni, si suppone che febbraio abbia sempre 28 giorni)
  [PSNLRI99B70C800M]
- Validità di nome e cognome (punto bonus, non vi viene spiegato esplicitamente ma è derivabile da come vengono generate le stringhe relative a nome e cognome)

Formato dei file di output *[codiciPersone.xml]*

Modello di output

```
<output>
  <persone numero="2">
    <persona id="0">
      <nome>Ilaria</nome>
      <cognome>Pasini</cognome>
      <sesso>F</sesso>
      <comune_nascita>Clusone</comune_nascita>
      <data_nascita>1999-04-24</data_nascita>
      <codice_fiscale>
        PSNLRI99D64C800M
      </codice_fiscale>
    </persona>
    <persona id="1">
      <nome>Chiara</nome>
      <cognome>Morra</cognome>
      <sesso>F</sesso>
      <comune_nascita>Onzo</comune_nascita>
      <data_nascita>1957-12-04</data_nascita>
      <codice_fiscale>
        ASSENTE
      </codice_fiscale>
    </persona>
  </persone>
...
```

(continua…)

```
...
  <codici>
    <invalidi numero="9">
      <codice>GDALKU87F92E2D2R</codice>
      <codice>ABCDEF78B57A024T</codice>
      <codice>LSHGDH76H08A167</codice>
      <codice>FSTPLA98M01B157F8</codice>
      <codice>STLSPR35T99B196T</codice>
      <codice>GRLMHI09B31A686K</codice>
      <codice>CEDSND06R53L820M</codice>
      <codice>MSAMDI8FM06E221C</codice>
      <codice>MSSMNL46C21E515K</codice>
    </invalidi>
    <spaiati numero="6">
      <codice>MNTPRZ24T67M178V</codice>
      <codice>PVNDSR19T68A293P</codice>
      <codice>BLDVNI90T45D585A</codice>
      <codice>MRCFRZ91P16F360I</codice>
      <codice>CTASNT00A59A630Z</codice>
      <codice>GCCSLD65T04D583R</codice>
    </spaiati>
  </codici>
</output>
```

Funzionalità aggiuntive

Scrittura del file in formato JSON.
Gli oggetti salvati nel file json dovranno avere gli stessi attributi mostrati nel file XML della slide precedente.
Modello di output:

```
{
  "persone": [
      {
        "nome": "Ilaria",
        "cognome": "Pasini",
        "sesso": "F",
        "comune_nascita": "Clusone",
        "data_nascita": "1999-04-24",
        "codice_fiscale": "PSNLRI99D64C800M"
      },
      {
        "nome": "Chiara",
        "cognome": "Morra",
        "sesso": "F",
        "comune_nascita": "Onzo",
        "data_nascita": "1957-12-04",
        "codice_fiscale": "ASSENTE"
      }
  ],

...
```

```
...
  "codici": {
    "invalidi": [
        "GDALKU87F92E2D2R",
        "ABCDEF78B57A024T",
        "LSHGDH76H08A167",
        "FSTPLA98M01B157F8",
        "STLSPR35T99B196T",
        "GRLMHI09B31A686K",
        "CEDSND06R53L820M",
        "MSAMDI8FM06E221C",
        "MSSMNL46C21E515K"
    ],
    "spaiati": [
        "MNTPRZ24T67M178V",
        "PVNDSR19T68A293P",
        "BLDVNI90T45D585A",
        "MRCFRZ91P16F360I",
        "CTASNT00A59A630Z",
        "GCCSLD65T04D583R"
    ]
  }
}
```

Note sulla valutazione

Oltre a valutare - come di consueto - il programma nel suo complesso, verranno valutate nello specifico le seguenti caratteristiche:

- Presenza di documentazione.
- Implementazione di lettura e scrittura dei dati su file XML.
- Correttezza della soluzione fornita dal programma
- Solidità del programma di fronte a scelte “peculiari” dell’utente (ossia: quanto è a prova di idiota il vostro codice? Avete previsto *tutti* i casi particolari?)
- Eventuali scelte di implementazione originali e interessanti che migliorino l’efficienza
- Utilizzo corretto di Git e GitHub.

Modalità di consegna

Una volta terminato il programma, esso dovrà essere caricato su piattaforma **GitHub** tramite l'account del responsabile del gruppo, all'interno di una nuova repository.

La **repository** dovrà chiamarsi "PgAr2023_NomeGruppo_CodiceFiscale", con il nome del vostro gruppo in *capitalized camel case* al posto della stringa "NomeGruppo". Tale repository dovrà contenere l'intero progetto *Eclipse* (o ancora meglio, di un altro IDE) del vostro gruppo con le eventuali dipendenze, in modo che poi si possa clonare e **sia già funzionante**. Il link deve essere inviato per email a *pgmarnaldo@googlegroups.com*, mettendo come oggetto il nome della repository.

L'intero progetto è da consegnare entro la data sotto riportata. Eventuali modifiche successive all'orario di consegna non verranno accettate.

Scadenza di consegna:
ore 23:59, 26/04/2023

Presentazione realizzata per lo Student Branch IEEE dell'Universitá degli Studi di Brescia, in occasione del Programma Arnaldo 2023

Si prega di non modificare o distribuire il contenuto di tale documento senza essere in possesso dei relativi permessi

corazzinamarco33@ieee.org
matteo.boniotti@ieee.org
stefano.agnelli@ieee.org
kibo@ieee.org

ieeesb.unibs.it

Grazie per l'attenzione