

# CODICE FISCALE

## Sommario

<b>CODICE FISCALE</b> .....	1
IL PROGETTO.....	2
PROGRAMMA .....	3
IL MAIN (dai dati al codice fiscale).....	3
Passaggio delle variabili alle funzioni .....	5
LE FUNZIONI (dai dati al codice fiscale) .....	6
Il calcolo del cognome .....	6
Il calcolo del nome .....	8
L'anno di nascita.....	9
Il mese di nascita .....	10
Calcolo del giorno di nascita .....	11
Il codice catastale.....	12
Il carattere di controllo .....	13
IL MAIN (dal codice fiscale ai dati).....	15
Le funzioni (dal codice fiscale ai dati) .....	16
Nome e cognome .....	16
Calcolo dell'anno.....	17
Il mese di nascita .....	18
Calcolo del giorno .....	19
Il codice catastale.....	20

## IL PROGETTO

Nella prima parte il progetto richiede di creare un programma in linguaggio C che, dati diversi input come: nome, cognome, data di nascita, il sesso e il comune di nascita, riesca a calcolare il codice fiscale. Per fare questo sappiamo che il codice fiscale è una sequenza di 16 caratteri alfanumerici nella quale i primi 15 contengono informazioni relative ai dati personali (citati in precedenza) nel seguente ordine:

- 3 caratteri alfabetici per il cognome
- 3 caratteri alfabetici per il nome
- 2 numeri per l'anno di nascita
- 1 carattere alfabetico per il mese di nascita
- 2 numeri per il giorno di nascita (N.B. +40 per le donne)
- 4 caratteri alfanumerici per il comune italiano o lo stato estero di nascita

Invece il sedicesimo carattere è un carattere di controllo trovato con uno specifico algoritmo sulla base degli altri quindici.

Ora un esempio per comprendere al meglio:

RSS BBR 69 C 48 F839 A

- RSS: caratteri indicativi del cognome
- BBR: caratteri indicativi del nome
- 69: caratteri indicativi dell'anno di nascita
- C: carattere indicativo del mese di nascita
- 48: caratteri indicativi del giorno di nascita e del sesso
- F839: caratteri indicativi del comune italiano di nascita
- A: carattere di controllo

Nella seconda parte, invece, è stato richiesto di implementare, tramite un apposito menù, la funzione inversa: dato il codice fiscale vengono estratti i singoli dati.

# PROGRAMMA

## IL MAIN (dai dati al codice fiscale)

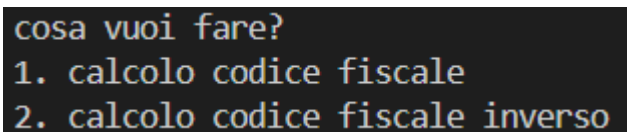
Appena viene avviato il programma, all'utente, viene subito chiesto cosa vuole fare

```
int main(void)
{
    int scegli;

    printf("cosa vuoi fare? \n");
    printf("1. calcolo codice fiscale \n");
    printf("2. calcolo codice fiscale inverso \n");

    scanf("%d", &scegli);
}
```

*Codice in linguaggio c*



*Terminale*

Ora analizziamo il programma nel caso l'utente scelga la prima opzione:

```
if (scegli == 1)
{
    char cognome[1], nome[1], anno[4], codice_fiscale[18];
    int mese;
    char giorno[3], sesso, comune[1];
    printf("inserisci il cognome:");
    scanf("%s", cognome);
    printf("inserisci il nome:");
    scanf("%s", nome);
    do
    {
        printf("inserisci l'anno di nascita:");
        scanf("%s", anno);
    } while (atoi(anno) > 2024);

    do
    {
        printf("inserisci il mese di nascita (in numero:");
        scanf("%d", &mese);
    } while (mese > 12 || mese < 1);

    printf("inserisci il giorno di nascita:");
    scanf("%s", giorno);
    printf("sesso (M o F):");
    scanf(" %c", &sesso);

    printf("inserisci comune di nascita: ");
    scanf("%s", comune);
}
```

*Codice in linguaggio c*

All'inizio sono dichiarate tutte le variabili necessarie nella quale vengono memorizzati tutti i dati inseriti dall'utente.

A questo punto viene chiesto all'utente di inserire il proprio cognome e nome memorizzandoli rispettivamente nell'array `cognome[]` e `nome[]`.

In seguito viene chiesto di inserire l'anno. Poiché l'anno è stato dichiarato come una stringa (per poi in seguito poterlo inserire all'interno dell'array: `codice_fiscale[]` senza

doversi preoccupare di inserire caratteri e valori interi COSA NON POSSIBILE), per verificare che esso sia valido si utilizza la funzione atoi (ascii to integer) che trasforma i caratteri in valori interi, per poi concludere con l’inserimento del mese, giorno, sesso e comune.

Esempio (questo esempio verrà riutilizzato in seguito):

```
inserisci il cognome:rossi
inserisci il nome:andrea
inserisci l'anno di nascita:2004
inserisci il mese di nascita (in numero):5
inserisci il giorno di nascita:15
sesso (M o F):M
inserisci comune di nascita: TORINO
```

*Terminale*

In questo momento  
quindi abbiamo:  
cognome[] = “rossi”  
nome[] = “andrea”  
anno[] = “2004”  
mese = 5  
giorno[] = “15”  
sesso = “M”  
comune[] = “TORINO”

## Passaggio delle variabili alle funzioni

Ora che questi dati sono stati memorizzati, il programma li passa a diverse funzioni che si occupano del calcolo dei singoli dati che in seguito vengono memorizzati nell'array `codice_fiscale[]`.

```
calcola_cognome(cognome, codice_fiscale);  
calcola_nome(nome, codice_fiscale);  
  
anno_nascita(anno, codice_fiscale);  
mese_nascita(mese, codice_fiscale);  
  
giorno_nascita(giorno, codice_fiscale, sesso);  
  
codice_catastale(comune, codice_fiscale);  
  
codice_controllo(codice_fiscale);
```

*Codice in linguaggio c*

A questo punto entriamo nel dettaglio e vediamo come si comportano le singole funzioni.

## LE FUNZIONI (dai dati al codice fiscale)

### Il calcolo del cognome

```
void calcola_cognome(char cognome[], char codice_fiscale[]) {  
  
    int consonanti = 0, vocali = 0;  
    int lunghezza = strlen(cognome);  
    char cons[20], voc[20], cognome2[lunghezza];  
  
    for(int i = 0; i < lunghezza; i++)  
    {  
        cognome2[i] = toupper(cognome[i]);  
        if(cognome2[i] == 'A' || cognome2[i] == 'E' || cognome2[i] == 'I' || cognome2[i] == 'O' || cognome2[i] == 'U'){  
            voc[vocali] = cognome2[i];  
            vocali++;  
        } else {  
            cons[consonanti] = cognome2[i];  
            consonanti++;  
        }  
    }  
}
```

*Codice in linguaggio c*

All'inizio della funzione sono state dichiarate diverse variabili:

- due variabili contatore (una per le vocali e una per le consonanti);
- una variabile lunghezza nella quale viene memorizzata la lunghezza del cognome con strlen (string length);
- tre array di tipo char, 2 per memorizzare le consonanti e le vocali trovate nel cognome e uno per memorizzare il cognome quando viene trasformato in maiuscolo con toupper (presente all'interno del ciclo for).

Dopo è presente una condizione per verificare che la lettera del cognome corrispondente all'indice "i" sia una vocale oppure una consonante. Nel caso sia una vocale (e quindi soddisfa la condizione presente all'interno dell'if), la lettera viene inserita nell'array "voc" altrimenti viene inserita nell'array "cons", incrementando anche i rispettivi contatori.

```
if(consonanti >= 3){  
    codice_fiscale[0] = cons[0];  
    codice_fiscale[1] = cons[1];  
    codice_fiscale[2] = cons[2];  
}  
  
if(consonanti == 2){  
    codice_fiscale[0] = cons[0];  
    codice_fiscale[1] = cons[1];  
    codice_fiscale[2] = voc[0];  
}  
  
if(consonanti == 1 && vocali == 2){  
    codice_fiscale[0] = cons[0];  
    codice_fiscale[1] = voc[0];  
    codice_fiscale[2] = voc[1];  
}  
  
if(consonanti == 1 && vocali == 1){  
    codice_fiscale[0] = cons[0];  
    codice_fiscale[1] = voc[0];  
    codice_fiscale[2] = 'X';  
}  
  
if(consonanti == 0 && vocali == 2){  
    codice_fiscale[0] = voc[0];  
    codice_fiscale[1] = voc[1];  
    codice_fiscale[2] = 'X';  
}  
}
```

Codice in linguaggio c

Una volta che è finito il ciclo for, e quindi è stato controllato tutto il cognome, in base al valore che hanno i contatori si entra all'interno di una delle seguenti condizioni, iniziando così a riempire le prime 3 posizioni del codice fiscale. Nelle ultime due condizioni poiché non ci sono abbastanza vocali e consonanti come terzo carattere viene inserita una 'X'.

Facendo riferimento all'esempio di prima:

ROSSI:

codice fiscale: RSS

Terminale

## Il calcolo del nome

```
void calcola_nome(char nome[], char codice_fiscale[]) {  
  
    int consonanti = 0, vocali = 0;  
    int lunghezza = strlen(nome);  
    char cons[20], voc[20], nome2[lunghezza];  
  
    for(int i = 0; i < lunghezza; i++)  
    {  
        nome2[i] = toupper(nome[i]);  
        if(nome2[i] == 'A' || nome2[i] == 'E' || nome2[i] == 'I' || nome2[i] == 'O' || nome2[i] == 'U'){  
            voc[vocali] = nome2[i];  
            vocali++;  
        }else {  
            cons[consonanti] = nome2[i];  
            consonanti++;  
        }  
    }  
}
```

Codice in linguaggio c

Nel calcolo del nome non ci sono troppe differenze rispetto al calcolo del cognome, le variabili dichiarate all'inizio sono le stesse così come il codice all'interno del ciclo for. Mentre alcune piccole differenze le troviamo nelle condizioni.

Per il nome il programma memorizza i tre caratteri nelle rispettive posizioni 3, 4 e 5.

```
if(consonanti >= 4){  
    codice_fiscale[3] = cons[0];  
    codice_fiscale[4] = cons[2];  
    codice_fiscale[5] = cons[3];  
}  
  
if(consonanti == 3){  
    codice_fiscale[3] = cons[0];  
    codice_fiscale[4] = cons[1];  
    codice_fiscale[5] = cons[2];  
}  
  
if(consonanti == 2){  
    codice_fiscale[3] = cons[0];  
    codice_fiscale[4] = cons[1];  
    codice_fiscale[5] = voc[0];  
}  
  
if(consonanti == 1 && vocali == 2){  
    codice_fiscale[3] = cons[0];  
    codice_fiscale[4] = voc[0];  
    codice_fiscale[5] = voc[1];  
}  
  
if(consonanti == 1 && vocali == 1){  
    codice_fiscale[3] = cons[0];  
    codice_fiscale[4] = voc[0];  
    codice_fiscale[5] = 'X';  
}  
  
if(consonanti == 0 && vocali == 2){  
    codice_fiscale[3] = voc[0];  
    codice_fiscale[4] = voc[1];  
    codice_fiscale[5] = 'X';  
}
```

Codice in linguaggio c

ANDREA:

codice fiscale: RS\$NDR

Terminale

RSS = ROSSI

NDR = ANDREA



## L'anno di nascita

Alla funzione “anno\_nascita”, sono state passate le variabili “anno” e “codice fiscale”.

```
void anno_nascita(char anno[],char codice_fiscale[]) {  
    codice_fiscale[6] = anno[2];  
    codice_fiscale[7] = anno[3];  
}
```

Poiché l'anno di nascita memorizza 4 caratteri e il codice fiscale richiede solo le ultime 2 cifre, si memorizzano nell'array

*Codice in linguaggio c*

codice\_fiscale[6] e [7] gli ultimi 2 caratteri dell'array anno[ ].

2004:

codice fiscale: RSSNDR04

*Terminale*

## Il mese di nascita

Alla funzione “mese\_nascita”, sono state passate le variabili “mese” e “codice fiscale”.

In seguito è stato utilizzato uno switch case alla quale è stata passata la variabile “mese”, in questo modo viene controllato quale numero (corrispondente al mese) è stato inserito e in base a quello viene memorizzata una lettera all'interno di codice\_fiscale[8].

**Tabella A – conversione del mese di nascita**

Gennaio = A	Maggio = E	Settembre = P
Febbraio = B	Giugno = H	Ottobre = R
Marzo = C	Luglio = L	Novembre = S
Aprile = D	Agosto = M	Dicembre = T

Tabella conversione del mese di nascita

```
void mese_nascita(int mese, char codice_fiscale[]) {  
    switch (mese) {  
        case 1:  
            codice_fiscale[8] = 'A';  
            break;  
        case 2:  
            codice_fiscale[8] = 'B';  
            break;  
        case 3:  
            codice_fiscale[8] = 'C';  
            break;  
        case 4:  
            codice_fiscale[8] = 'D';  
            break;  
        case 5:  
            codice_fiscale[8] = 'E';  
            break;  
        case 6:  
            codice_fiscale[8] = 'H';  
            break;  
    }  
}
```

Codice in linguaggio c

```
        case 7:  
            codice_fiscale[8] = 'L';  
            break;  
        case 8:  
            codice_fiscale[8] = 'M';  
            break;  
        case 9:  
            codice_fiscale[8] = 'P';  
            break;  
        case 10:  
            codice_fiscale[8] = 'R';  
            break;  
        case 11:  
            codice_fiscale[8] = 'S';  
            break;  
        case 12:  
            codice_fiscale[8] = 'T';  
            break;  
        default:  
            break;  
    }  
}
```

Codice in linguaggio c

Mese: 5 (=maggio)

codice fiscale: RSSNDR04E

Terminale

## Calcolo del giorno di nascita

Alla funzione “giorno\_nascita”, sono state passate le variabili “giorno”, “codice\_fiscale” e “sesso”.

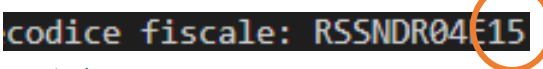
```
void giorno_nascita(char giorno[], char codice_fiscale[], char sesso) {  
    if(sesso == 'M'){  
        codice_fiscale[9] = giorno[0];  
        codice_fiscale[10] = giorno[1];  
    }else if (sesso == 'F') {  
        int giorno_f = atoi(giorno) + 40;  
        sprintf(codice_fiscale + 9, "%02d", giorno_f);  
    }  
}
```

*Codice in linguaggio c*

Il programma verifica se il carattere inserito è “M” (maschio) oppure “F” (femmina). Nel primo caso il giorno verrà memorizzato così com’è stato inserito in codice\_fiscale[9] e [10], mentre nel secondo caso (carattere inserito “F”), verrà dichiarata una nuova variabile nella quale si memorizza il giorno aumentato di 40, (poiché la variabile giorno è un’array di carattere, è stata utilizzata la funzione atoi, già citata in precedenza p. 4). In seguito è stata utilizzata un’altra funzione “sprintf” per convertire nuovamente il valore in stringa e poi inserirlo in codice\_fiscale[9] e [10].

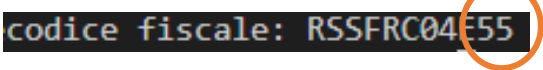
Sesso: “M”

Giorno: 15

  
*Terminale*

Sesso: “F”

Giorno: 15

  
*Terminale*

## Il codice catastale

Alla funzione “codice\_catastale”, sono state passate le variabili “comune” e “codice\_fiscale”.

```
void codice_catastale(char comune[], char codice_fiscale[]) {  
  
    char riga[L];  
    FILE *fp;  
  
    fp = fopen("codici_catastali.txt", "r");  
  
    if(fp == NULL){  
        printf("errore apertura del file");  
        return;  
    }  
  
    while (fgets(riga, L, fp) != NULL) {  
  
        if(strstr(riga, comune) != NULL){  
            strncpy(codice_fiscale + 11, riga, 4);  
            fclose(fp);  
        }  
    }  
  
    fclose(fp);  
}
```

Codice in linguaggio c

Inizialmente è stata creata una variabile `riga` e anche una variabile puntatore che punta al file nella quale sono presenti i vari codici catastali.

Come prima operazione viene aperto il file “codici\_catastali.txt” in modalità lettura “r”.

In seguito il puntatore `fp` restituisce un valore. Se esso è uguale a “NULL” significa che non è stato possibile aprire il file

e il programma viene arrestato con “return”.

Il while, invece, viene utilizzato per scorrere tutte le righe del file e ad ogni riga viene confrontato con la funzione `strstr`, una funzione che confronta due stringhe una più grande (in questo caso “`riga[]`”) e una più piccola, cioè una sottostringa, (in questo caso “`comune[]`”) e verifica che sia diverso da “NULL”. Se è diverso e quindi la sottostringa è stata trovata, allora, con “`strncpy`” verranno copiati i primi 4 caratteri del file (in cui è presente il codice del comune) a partire dalla posizione 11 dell’array e infine verrà chiuso il file.

Ecco come si presentano le righe all’interno del file:



File `codici_catastali.txt`

i primi 4 caratteri corrispondono al comune e in seguito è presente il nome del comune.

Torino:

codice fiscale: RSSNDR04E15L219

Terminale

## Il carattere di controllo

Alla funzione di “codice\_controllo”, poiché il carattere viene calcolato sulla base dei caratteri precedenti, è stata passata solamente la variabile “codice\_fiscale”.

```
void codice_controllo(char codice_fiscale[]) {  
  
    int r = 0, s = 0;  
    char lettera_controllo;  
  
    for (int i = 0; i < 15; i++) {  
        if (i % 2 == 0) {  
            switch (codice_fiscale[i]) {  
                case 'A': case '0': r += 1; break;  
                case 'B': case '1': r += 0; break;  
                case 'C': case '2': r += 5; break;  
                case 'D': case '3': r += 7; break;  
                case 'E': case '4': r += 9; break;  
                case 'F': case '5': r += 13; break;  
                case 'G': case '6': r += 15; break;  
                case 'H': case '7': r += 17; break;  
                case 'I': case '8': r += 19; break;  
                case 'J': case '9': r += 21; break;  
                case 'K': r += 2; break;  
                case 'L': r += 4; break;  
                case 'M': r += 18; break;  
                case 'N': r += 20; break;  
                case 'O': r += 11; break;  
                case 'P': r += 3; break;  
                case 'Q': r += 6; break;  
                case 'R': r += 8; break;  
                case 'S': r += 12; break;  
                case 'T': r += 14; break;  
                case 'U': r += 16; break;  
                case 'V': r += 10; break;  
                case 'W': r += 22; break;  
                case 'X': r += 25; break;  
                case 'Y': r += 24; break;  
                case 'Z': r += 23; break;  
            }  
        }  
    }  
}
```

Codice in linguaggio c

All’inizio della funzione sono state dichiarate 2 variabili di tipo intero e 1 di tipo char:

- “r” inizializzata con il valore 0;
- “s” inizializzata con il valore 0;
- “lettera\_controllo” nella quale verrà memorizzato il carattere finale.

Ogni carattere è associato ad un numero. I numeri in seguito vengono addizionati e divisi per 26. Il resto della divisione, poi, viene convertito in un carattere ed esso sarà il carattere di controllo.

Nel codice possiamo vedere un ciclo for che scorre tutte le lettere del codice fiscale, poiché in base alla posizione il carattere cambia valore (se il carattere si trova in posizione pari avrà un determinato valore altrimenti un altro), è presente una condizione che verifica se l’indice “i” è pari oppure è dispari, e verrà passato allo

switch case il carattere del codice fiscale che corrisponde a tale indice. In seguito verrà confrontato con quelli presenti all’interno dei “case” e verrà sommato tale valore alla variabile “r”.

```
} else {  
    switch (codice_fiscale[i]) {  
        case 'A': case '0': r += 0; break;  
        case 'B': case '1': r += 1; break;  
        case 'C': case '2': r += 2; break;  
        case 'D': case '3': r += 3; break;  
        case 'E': case '4': r += 4; break;  
        case 'F': case '5': r += 5; break;  
        case 'G': case '6': r += 6; break;  
        case 'H': case '7': r += 7; break;  
        case 'I': case '8': r += 8; break;  
        case 'J': case '9': r += 9; break;  
        case 'K': r += 10; break;  
        case 'L': r += 11; break;  
        case 'M': r += 12; break;  
        case 'N': r += 13; break;  
        case 'O': r += 14; break;  
        case 'P': r += 15; break;  
        case 'Q': r += 16; break;  
        case 'R': r += 17; break;  
        case 'S': r += 18; break;  
        case 'T': r += 19; break;  
        case 'U': r += 20; break;  
        case 'V': r += 21; break;  
        case 'W': r += 22; break;  
        case 'X': r += 23; break;  
        case 'Y': r += 24; break;  
        case 'Z': r += 25; break;  
    }  
}
```

*Codice in linguaggio c*

La variabile s poi verrà passata ad un altro switch case che in base al resto memorizza il carattere corrispondente nella variabile “lettera\_controllo”.

Per poi terminare il programma memorizzando il carattere di controllo in “codice\_fiscale[15]” e stampando il codice fiscale completo a schermo.

Stessa cosa varrà per i caratteri in posizione dispari (riportati nell’immagine a sinistra).

In seguito nella variabile “s”, verrà memorizzato il resto della divisione per 26.

```
s = r % 26;
```

```
switch (s) {  
    case 0: lettera_controllo = 'A'; break;  
    case 1: lettera_controllo = 'B'; break;  
    case 2: lettera_controllo = 'C'; break;  
    case 3: lettera_controllo = 'D'; break;  
    case 4: lettera_controllo = 'E'; break;  
    case 5: lettera_controllo = 'F'; break;  
    case 6: lettera_controllo = 'G'; break;  
    case 7: lettera_controllo = 'H'; break;  
    case 8: lettera_controllo = 'I'; break;  
    case 9: lettera_controllo = 'J'; break;  
    case 10: lettera_controllo = 'K'; break;  
    case 11: lettera_controllo = 'L'; break;  
    case 12: lettera_controllo = 'M'; break;  
    case 13: lettera_controllo = 'N'; break;  
    case 14: lettera_controllo = 'O'; break;  
    case 15: lettera_controllo = 'P'; break;  
    case 16: lettera_controllo = 'Q'; break;  
    case 17: lettera_controllo = 'R'; break;  
    case 18: lettera_controllo = 'S'; break;  
    case 19: lettera_controllo = 'T'; break;  
    case 20: lettera_controllo = 'U'; break;  
    case 21: lettera_controllo = 'V'; break;  
    case 22: lettera_controllo = 'W'; break;  
    case 23: lettera_controllo = 'X'; break;  
    case 24: lettera_controllo = 'Y'; break;  
    case 25: lettera_controllo = 'Z'; break;  
}
```

*Codice in linguaggio c*

```
codice_fiscale[15] = lettera_controllo;  
printf("codice fiscale: %s \n", codice_fiscale);  
}
```

*Codice in linguaggio c*

```
codice fiscale: RSSNDR04E15L219L
```

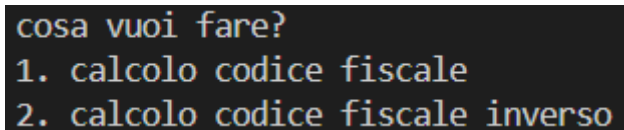
*Terminale*

## IL MAIN (dal codice fiscale ai dati)

Ora che abbiamo concluso la prima parte e siamo riusciti a trovare il codice fiscale, guardiamo come fare l'opposto: Inserendo il codice fiscale il programma estrae i dati.

```
} else {  
    char codice_fiscale_inverso[16];  
  
    printf("inserisci il codice fiscale: ");  
    scanf(" %s", codice_fiscale_inverso);  
  
    nome_cognome_inverso(codice_fiscale_inverso);  
    anno_inverso(codice_fiscale_inverso);  
    mese_inverso(codice_fiscale_inverso);  
    giorno_inverso(codice_fiscale_inverso);  
    codice_catastale_inverso(codice_fiscale_inverso);  
}  
  
return 0;  
}
```

*Codice in linguaggio c*

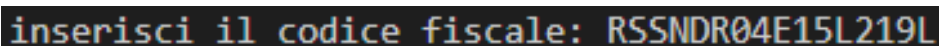


```
cosa vuoi fare?  
1. calcolo codice fiscale  
2. calcolo codice fiscale inverso
```

*Terminale*

Una volta avviato il programma e scelto la seconda opzione (“calcolo del codice fiscale inverso”) il programma dichiarerà la variabile

“codice\_fiscale\_inverso[16]” e verrà chiesto di inserire il codice fiscale, per poi passare la variabile a diverse funzioni che ora vedremo nel dettaglio.



```
inserisci il codice fiscale: RSSNDR04E15L219L
```

*Terminale*

## Le funzioni (dal codice fiscale ai dati)

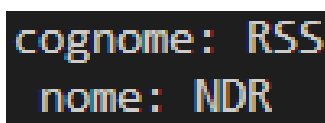
### Nome e cognome

```
void nome_cognome_inverso(char codice_fiscale_inverso[]){  
  
    char Cognome_inverso[4], Nome_inverso[4];  
  
    Cognome_inverso[3] = '\0';  
    Nome_inverso[3] = '\0';  
  
    for(int i = 0; i < 3; i++)  
        Cognome_inverso[i] = codice_fiscale_inverso[i];  
  
    for(int i = 3; i < 6; i++)  
        Nome_inverso[i - 3] = codice_fiscale_inverso[i];  
  
    printf("cognome: %s \n nome: %s \n ", Cognome_inverso, Nome_inverso);  
}
```

*Codice in linguaggio c*

Nella prima funzione inizialmente vengono dichiarati 2 array in cui alla posizione 3 di ognuno vengono inizializzate con '\0' per far sì che terminino correttamente.

In seguito troviamo un primo ciclo for che memorizza i primi 3 caratteri del codice fiscale nell'array "Cognome\_inverso". Il secondo for farà la stessa cosa ma dal momento che i 3 caratteri del nome si trovano dalla posizione 3 alla 5 nell'array "Nome\_inverso[i]" viene aggiunto un -3 per far sì che l'array parta a memorizzare i caratteri dalla posizione 0. Per poi concludere stampando i 3 caratteri per il cognome e i 3 per il nome.



*Terminale*

N.B. Poiché non è possibile trovare il cognome e il nome avendo solo 3 caratteri ciascuno verranno stampati semplicemente i 3 caratteri corrispondenti.



## Calcolo dell'anno

```
void anno_inverso(char codice_fiscale_inverso[]) {  
  
    char Anno_inverso[5];  
  
    Anno_inverso[0] = codice_fiscale_inverso[6];  
    Anno_inverso[1] = codice_fiscale_inverso[7];  
  
    if(atoi(Anno_inverso) < 25){  
        Anno_inverso[0] = '2';  
        Anno_inverso[1] = '0';  
        Anno_inverso[2] = codice_fiscale_inverso[6];  
        Anno_inverso[3] = codice_fiscale_inverso[7];  
        Anno_inverso[4] = '\0';  
    } else {  
        Anno_inverso[0] = '1';  
        Anno_inverso[1] = '9';  
        Anno_inverso[2] = codice_fiscale_inverso[6];  
        Anno_inverso[3] = codice_fiscale_inverso[7];  
        Anno_inverso[4] = '\0';  
    }  
  
    printf("anno: %s \n", Anno_inverso);  
}
```

*Codice in linguaggio c*

All'inizio della funzione è stato dichiarato un array "Anno\_inverso" di tipo char.

Poiché nel codice fiscale i caratteri dell'anno si trovano alla posizione 6 e 7 vengono memorizzate nell'array dichiarato all'inizio nelle posizioni 0 e 1.

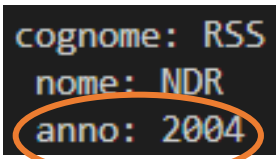
In seguito per sapere se si è nati prima del 2000 o dopo è stata trasformata la stringa in un valore intero con la funzione atoi e nel caso il numero sia minore di 25 allora in "Anno\_inverso" nelle posizioni 0 e 1 vengono

memorizzati i caratteri 2 e 0 e in

seguito i 2 caratteri presenti nel codice\_fiscale\_inverso[6] e [7] vengono memorizzati in Anno\_inverso[2] e [3] così facendo alla fine verrà stampato l'anno di nascita intero.

Invece se viene inserito un codice fiscale nella quale i 2 caratteri corrispondenti all'anno sono maggiori di 25, nelle prime 2 posizioni di "Anno\_inverso" vengono memorizzati i caratteri 1 e 9.

Infine con "printf" viene stampato l'array con l'anno completo.



cognome: RSS  
nome: NDR  
anno: 2004

*Terminale*

## Il mese di nascita

```
void mese_inverso(char codice_fiscale_inverso[]) {  
    switch (codice_fiscale_inverso[8])  
    {  
        case 'A':  
            printf("mese: gennaio \n");  
            break;  
        case 'B':  
            printf("mese: febbraio \n");  
            break;  
        case 'C':  
            printf("mese: marzo \n");  
            break;  
        case 'D':  
            printf("mese: aprile \n");  
            break;  
        case 'E':  
            printf("mese: maggio \n");  
            break;  
        case 'H':  
            printf("mese: giugno \n");  
            break;  
        case 'L':  
            printf("mese: luglio \n");  
            break;  
    }
```

*Codice in linguaggio c*

```
        case 'M':  
            printf("mese: agosto \n");  
            break;  
        case 'P':  
            printf("mese: settembre \n");  
            break;  
        case 'R':  
            printf("mese: ottobre \n");  
            break;  
        case 'S':  
            printf("mese: novembre \n");  
            break;  
        case 'T':  
            printf("mese: dicembre \n");  
            break;  
        default:  
            break;  
    }
```

*Codice in linguaggio c*

Nella funzione “mese\_inverso” è presente uno switch case alla quale è stata passata la variabile “codice\_fiscale\_inverso[8]” (la posizione in cui è scritta la lettera corrispondente al mese). La lettera viene confrontata con tutti i caratteri all’interno dei case e una volta trovata viene stampato a schermo il mese corrispondente.

```
cognome: RSS  
nome: NDR  
anno: 2004  
mese: maggio
```

*Terminale*

## Calcolo del giorno

All'inizio della funzione "giorno\_inverso", viene dichiarata una variabile giorno di tipo intero che è stata inizializzata con il valore di "atoi di codice\_fiscale\_inverso[9]" nella quale è presente il giorno.

```
void giorno_inverso(char codice_fiscale_inverso[]) {  
    int giorno = atoi(codice_fiscale_inverso[9]);  
    if (giorno >= 1 && giorno <= 31) {  
        printf("giorno: %02d\n", giorno);  
    } else if (giorno >= 41 && giorno <= 71) {  
        giorno -= 40;  
        printf("giorno: %02d\n", giorno);  
    }  
}
```

Dal momento che nel codice di controllo il giorno per le donne viene aumentato di 40 (e quindi i giorni partono da 41 e arrivano a 71), nel codice è presente una condizione che verifica se il valore della variabile "giorno" è compreso tra 1 e 31 oppure tra 41 e 71.

*Codice in linguaggio c*

Nel primo caso il giorno viene stampato così com'è scritto nel codice fiscale, mentre, nel secondo caso (nel quale il codice fiscale corrisponde ad una donna), il giorno viene prima diminuito di 40 per poi essere stampato a schermo.

Primo caso ("M"):

```
inserisci il codice fiscale: RSSNDR04151219L  
cognome: RSS  
nome: NDR  
anno: 2004  
mese: maggio  
giorno: 15
```

*Terminale*

Secondo caso ("F"):

```
inserisci il codice fiscale: RSSFRC04551219L  
cognome: RSS  
nome: FRC  
anno: 2004  
mese: maggio  
giorno: 15
```

*Terminale*

## Il codice catastale

Per concludere come ultima funzione troviamo quella per identificare il comune di nascita in base al codice catastale.

```
void codice_catastale_inverso(char codice_fiscale_inverso[]) {  
  
    char riga[L];  
    FILE *fp;  
  
    fp = fopen("codici_catastali.txt", "r");  
  
    if (fp == NULL) {  
        printf("Errore durante l'apertura del file \n");  
        return;  
    }  
  
    char cod_catastale[5];  
    strncpy(cod_catastale, codice_fiscale_inverso + 11, 4);  
    cod_catastale[4] = '\0';  
  
    while (fgets(riga, L, fp) != NULL) {  
        if (strncmp(riga, cod_catastale, 4) == 0) {  
            printf("comune di nascita: %s\n", riga + 5);  
            fclose(fp);  
            return;  
        }  
    }  
  
    printf("nessun comune trovato \n");  
    fclose(fp);  
}
```

*Codice in linguaggio c*

Per iniziare così come nella funzione per il codice catastale vista in precedenza, sono state dichiarate un array “riga[]” e un puntatore al file “codici\_catastali.txt”.

Subito dopo, il programma apre il file in modalità lettura “r” e il puntatore fp restituisce un valore. Se esso è uguale a “NULL” viene stampato a schermo un messaggio di errore e il programma viene fermato con return. Nel caso l’apertura del file vada a buon fine viene dichiarato un nuovo array “cod\_catastale[]”, nella quale viene copiato il codice catastale con “strncpy”.

In seguito il while scorre tutte le righe del file fin quando non viene trovato il codice del comune (oppure fin quando non arriva alla fine del file).

Ad ogni ciclo il programma controlla i primi 4 caratteri di “riga[]” in cui è presente il codice catastale. Se il codice viene confrontato ed è stato trovato verrà stampato a schermo l’array riga partendo dal carattere 5 (in cui è presente il nome del comune evitando così di stampare il codice catastale).

Infine il file viene chiuso e il programma viene arrestato con return.

```
inserisci il codice fiscale: RSSNDR04E15L219L
cognome: RSS
nome: NDR
anno: 2004
mese: maggio
giorno: 15
comune di nascita: TORINO (TO) TO
```

*Terminale*