Esercizio 1

- Realizzare in linguaggio C un semplice editor di testo. Il programma dovrà:
  1. Chiedere all'utente il nome del file su cui salvare il testo. Se il nome è "" (stringa vuota) usare come nome del file `output.txt`.
  2. Chiedere all'utente ciclicamente il testo da scrivere nel file.
  3. Se il testo inserito è solo la parola "END", il programma deve chiudere il file e terminare l'esecuzione.

Esercizio 2

- Basandosi sull'esercizio precedente, si desidera modificare l'output del programma: ogni linea scritta nel file generato dal programma deve essere preceduta:
  - Dal numero della linea stessa, preceduto dal numero di zeri necessari affinché sia formato da almeno tre cifre;
  - Dal numero di caratteri della linea stessa.

Esercizio 2 – esempio

- Esempio di output su file del programma:

```
001 35 Nel mezzo del cammin di nostra vita
002 32 mi ritrovai per una selva oscura
003 32 che' la diritta via era smarrita
004 38 Ahi quanto a dir qual era e' cosa dura
005 36 esta selva selvaggia e aspra e forte
006 31 che nel pensier rinova la paura
007 36 Tant'e' amara che poco e' piu' morte
008 38 ma per trattar del ben ch'i' vi trovai
009 39 diro' de l'altre cose ch'i' v'ho scorte
```

Esercizio 3

- Una nota industria di telecomunicazioni ha scelto di adottare il programma dell'esercizio 1 per il suo sistema internazionale di telegrammi, a patto che introduciate due modifiche:
  1. Sul file di output si desidera che al posto dei punti (".") venga scritta la parola "STOP".
  2. Il numero di caratteri di ogni linea, calcolato prima di inserire gli STOP, venga salvato anche sul file binario `numcaratteri.dat`.

Esercizio 3 - esempio

- Esempio di input inserito dall'utente:

```
Messaggio urgente. Ricordarsi comperare pane et
  burro.
Attenzione. Virare babordo gradi nove aut
  prepararsi all'impatto.
END
```

- Sul file di output il programma deve scrivere:

```
Messaggio urgente STOP Ricordarsi comperare pane
  et burro STOP
Attenzione STOP Virare babordo gradi nove aut
  prepararsi all'impatto STOP
```

- Sul file binario di output `numcaratteri.dat` invece devono essere presenti i valori (in formato binario):

```
54 67
```

Esercizio 4

- Scrivere un programma che:
  - Si scriva la funzione:

    `unsigned int succprimo(unsigned int n);`

    che dato un intero n, restituisca il numero primo x>=n, usando come riferimento il file `primi.dat` (generato come da punto seguente) – o 0 se il file `primi.dat` non contiene un valore adatto.
  - Memorizzi in un array di 100 interi senza segno i primi 100 numeri primi e li salvi sul file binario `primi.dat`. I numeri primi devono essere generati usando ripetutamente la funzione:

    `unsigned int cercaprimo(unsigned int *v, unsigned int len)`

    che dato un array v di `len` numeri primi, restituisce il successivo numero primo.
    - Suggerimento: il primo numero primo è 2. I numeri primi successivi sono tutti quei numeri non divisibili per i numeri primi che lo precedono
  - Chieda all'utente un numero intero senza segno e ne stampi il primo successivo utilizzando la funzione di appoggio `succprimo`.
  - Chieda all'utente se desidera cercare un altro numero primo, e termini in caso negativo.

Esercizio 5

- Si realizzi un programma C per gestire un catalogo di condensatori.
- Si scriva una funzione
  `float energia (int tensione, float capacita)`
  che calcola l'energia conservata in un condensatore, utilizzando la formula $E = (CV^2)/2$
- Si crei la struttura **`condensatore`** contenente:
  - Il campo `codice` (array di 10 caratteri) che contiene codice univoco che identifica il condensatore;
  - Il campo `tipo` (array di 20 caratteri) per contenere il tipo di condensatore (al tantalio, ceramico, ecc.);
  - Il campo `tensione` (intero) in V;
  - Il campo `capacita` (float) in µF;
  - Il campo `energia` (float) in µJ.
- Si leggano dal file `condensatori.txt` i dati di condensatori e li si usi per popolare un array di strutture `condensatore`.
- In particolare, ogni linea del file contiene:
  - Il codice del componente;
  - Il tipo di condensatore;
  - La tensione di lavoro in Volt indicata come numero intero;
  - La capacità del condensatore in µF come numero reale.
    - Per calcolare l'energia contenuta, si utilizzi la funzione `energia` precedentemente realizzata.
- Si salvi in formato binario l'array sul file `catalogocond.bin`.

Esercizio 5 – segue

- Scrivere la funzione
  `void stampacondensatore(condensatore c)`
  che stampa tutte le informazioni relative a un condensatore
- Scrivere la funzione
  `void cercapertipo()`
  che chiede all'utente un tipo di condensatore, apre il file `catalogocond.bin` e stampa tutti i condensatori di quel tipo usando la funzione `stampacondensatore`.
- Scrivere le funzioni
  `void cercatensione()`
  `void cercacapacita()`
  `void cercaenergia()`
  che chiedono all'utente una tensione, una capacita o un'energia e stampano su schermo le informazioni relative ai condensatori con valore maggiore o uguale a quello richiesto (usando la funzione `stampacondensatore`), cercandoli nel file `catalogocond.bin`.
- Chiedere all'utente che tipo di ricerca effettuare nel catalogo e, in base alla scelta, richiamare una delle funzioni precedentemente definite.

Esercizio 5 – segue

- Riassumendo, il main del programma deve:
  - Accedere al file `condensatori.txt`, usarlo per popolare un array di strutture condensatore e salvarlo in formato binario sul file `catalogocond.bin`.
  - Chiedere all'utente in base a quale criterio cercare i condensatori, e in base a questa scelta richiamare un'opportuna funzione di ricerca.
  - Effettuare ciclicamente la richiesta, fino a quando l'utente non decide di terminare l'esecuzione del programma.