

# A.A. 2019/2020 - Elaborato 1

*Trovare la parola più lunga all'interno di una frase e cifrare/decifrare un testo con il cifrario di Vigenère, usando come chiave la parola trovata.*

**Input:** due stringhe (frase e testo cifrato o da cifrare), entrambi array di BYTE terminanti con 0; un BYTE (0 se il testo è da cifrare, 1 se è da decifrare).

**Output:** due stringhe (array di BYTE da terminare con il valore 0) contenenti la parola più lunga e il testo cifrato (o decifrato).

## Esempi di casi importanti da verificare:

- Frase contenente una sola parola.
- Testo da cifrare/decifrare con lunghezza nulla.
- Chiave più corta (o più lunga) del testo.

## N.B.

- Dettagli sul cifrario di Vigenère:  
[https://it.wikipedia.org/wiki/Cifrario\\_di\\_Vigen%C3%A8re](https://it.wikipedia.org/wiki/Cifrario_di_Vigen%C3%A8re)
- Si supponga che sia il testo da cifrare che la parola chiave non contengano mai caratteri con codice ASCII > 127.
- Si supponga che la frase contenga sempre almeno una parola.

## Scheletro da utilizzare per il programma:

```
void main()
{
    // Input
    char frase[1024] = "Cantami, o Diva, del Pelide Achille l'ira funesta che infiniti addusse";
    char testo[1024] = "Questo e' il testo da cifrare";
    char decifra = 0; // 0 = cifra; 1 = decifra

    // Output
    char parolaMax[100]; //La parola più lunga
    char risultato[1024]; //Il testo cifrato/decifrato

    // Blocco assembler
    __asm
    {

    // Stampa a video
    printf("Frase: %s\n", frase);
    printf("Testo: %s\n", testo);
    printf("Parola piu' lunga: %s\n", parolaMax);
    // Stampa su video testo cifrato
    printf("Risultato: %s\n", risultato);
    printf("Risultato (in esadecimale):\n");
    // N.B. Nella stringa precedente alcuni caratteri possono non essere stampabili:
    // crea anche una stringa con tutti i caratteri come codici ascii esadecimali
    char* t = risultato;
    while (*t)
    {
        printf("\\x%02X", (unsigned char)(*t++));
    }
}
```

# A.A. 2019/2020 - Elaborato 2

*Dato un insieme di  $n$  punti  $m$ -dimensionali, scomporre in fattori primi il prodotto degli indici del punto più vicino e di quello più lontano rispetto a un punto  $p$ .*

**Input:** due array di WORD (signed) contenenti rispettivamente le coordinate del punto  $p$  e degli  $n$  punti appartenenti all'insieme (es.  $m=3$ ,  $n=2$ ,  $\text{pointset[]}=\{x_1, y_1, z_1, x_2, y_2, z_2\}$ ). Due DWORD (unsigned) contenenti il numero di dimensioni di ogni punto ( $m$ ) e il numero di punti ( $n$ ) contenuti nell'insieme.

**Output:** due WORD (unsigned) per gli indici del punto più vicino e del punto più lontano a  $p$ , una WORD (unsigned) per il prodotto dei due indici, un vettore di DWORD (unsigned) per i fattori primi e una DWORD (unsigned) per il numero di fattori trovati.

## N.B.

- Si supponga che l'insieme contenga sempre almeno un punto.
- I valori delle coordinate dei vari punti sono compresi nell'intervallo  $[-100;100]$ .
- Per calcolare la distanza euclidea tra due punti potete consultare il link seguente: [http://it.wikipedia.org/wiki/Distanza\\_euclidea](http://it.wikipedia.org/wiki/Distanza_euclidea).
- Per trovare la distanza minima e massima non è necessario calcolare la radice quadrata.
- I fattori primi devono essere riportati nell'array in ordine crescente.

## Scheletro da utilizzare per il programma:

```
void main()
{
    // Input
    short pointP[] = { 0,0 }; //Punto p
    short pointset[] = { 42, -53, 96, 78, 1, 1, 100, 100 }; //Array di short contenente
                                                         //le coordinate degli n punti
    unsigned int m = 2; //Numero delle dimensioni di
                       //ogni punto
    unsigned int n = sizeof(pointset) / (sizeof(pointset[0]) * m); //Numero di punti

    // Output
    unsigned short indiceMin; //Indice del punto più vicino a p
    unsigned short indiceMax; //Indice del punto più lontano da p
    unsigned int prodotto; //Prodotto degli indici del punto più vicino e del punto
                           //più lontano
    unsigned int fattori[100]; //Fattori primi trovati
    unsigned int numFattori; // Numero dei fattori trovati

    // Blocco assembler
    __asm
    {

    }

    // Stampa su video
    printf("Indice del punto piu' vicino a p: %d\n", indiceMin);
    printf("Indice del punto piu' lontano da p: %d\n", indiceMax);
    printf("Prodotto degli indici: %d\n", prodotto);
    printf("Numero di fattori primi=%d\n", numFattori);
    for (unsigned int i = 0; i < numFattori; i++)
        printf("%d\n", fattori[i]);
}
```

# A.A. 2019/2020 - Elaborato 3

*Data una sequenza di bit, restituire la lunghezza media (arrotondata all'intero più vicino), minima e massima delle sotto-sequenze di bit (contigui) a 0 e a 1.*

**Input:** un array di BYTE da considerare come una sequenza di bit; una WORD (il numero totale di bit).

**Output:** sei WORD (unsigned) per la lunghezza media, minima e massima delle sotto-sequenze di 0 e 1.

**Esempi di casi importanti da verificare:**

- Array senza sequenze di bit a 0.
- Array senza sequenze di bit a 1.

**N.B.**

- viene specificata in input la lunghezza in *bit*: può quindi accadere che l'ultimo BYTE dell'array sia utilizzato solo in parte.
- I *bit* all'interno di ciascun BYTE devono essere analizzati dal meno significativo al più significativo.

Scheletro da utilizzare per il programma:

```
void main()
{
    // Input
    unsigned char vet[] = { 0xBC,0xFF,0x01 };    //Sequenza di bit
    unsigned short int len = 18;                  // Lunghezza (in bit) della sequenza

    // Output
    unsigned short int minLungSeq0;               //Lunghezza minima sotto-sequenza di 0
    unsigned short int maxLungSeq0;               //Lunghezza massima sotto-sequenza di 0
    unsigned short int mediaLungSeq0;             //Media arrotondata sotto-sequenze di 0
    unsigned short int minLungSeq1;               //Lunghezza minima sotto-sequenza di 1
    unsigned short int maxLungSeq1;               //Lunghezza massima sotto-sequenza di 1
    unsigned short int mediaLungSeq1;             // Media arrotondata sotto-sequenze di 1

    // Blocco assembler
    __asm
    {

    }

    // Stampa su video
    printf("Lunghezza minima delle sotto-sequenze di 0: %d\n", minLungSeq0);
    printf("Lunghezza massima delle sotto-sequenze di 0: %d\n", maxLungSeq0);
    printf("Lunghezza Media delle sotto-sequenze di 0: %d\n", mediaLungSeq0);
    printf("Lunghezza minima delle sotto-sequenze di 1: %d\n", minLungSeq1);
    printf("Lunghezza massima delle sotto-sequenze di 1: %d\n", maxLungSeq1);
    printf("Lunghezza Media delle sotto-sequenze di 1: %d\n", mediaLungSeq1);
}
```