Laboratorio 6

Nella propria home directory creare una sottodirectory chiamata es06,
in cui metteremo tutti i file C di oggi.

Esercizi su procedure e funzioni

Nel seguito, quando si dice "riceve un array" o qualcosa di equivalente,
si intende *SEMPRE* che venga passata anche la lunghezza dell'array come
parametro, anche se non viene esplicitamente detto.

Inoltre si richiede che, oltre alla funzione o procedura che e' stata richiesta,
scriviate anche un main che usi tale funzione e ne dimostri *ogni*
funzionalita'.

Non e' invece necessario che scriviate le funzioni in file separati, potete lavorare
per ogni esercizio su un unico file.

Per ultimo, se con l'esercizio viene fornita la firma(o prototipo) della funzione/procedura da
scrivere, si richiede che la soluzione rispetti questa firma in ogni dettaglio (nome
funzione, tipo di ritorno, lista di argomenti).

Ricordate che una procedura e' una funzione con tipo di ritorno void.

Infine in tutti gli esercizi che seguono siete invitati, quando e' opportuno, ad
applicare gli schemi di ricerca certa, incerta e verifica di una proprieta'
visti a lezione.

Per quanto riguarda l'uso dei booleani utilizzare

```
#DEFINE FALSE 0
#DEFINE TRUE  1
```

come visto a lezione.

Esercizi introduttivi

1) Scrivere una procedura che riceva un array di caratteri e la sua lunghezza
e lo stampi a video

2) Scrivere una funzione che riceva un array di interi e ritorni l'ultimo elemento
dell'array.

3) Scrivere una procedura che, ricevuto un array di interi, ne azzeri
tutti gli elementi.

4) Scrivere una funzione che, ricevuto un array di interi, ritorni la somma di
tutti gli elementi.

5) Scrivere una funzione che, ricevuto un array di caratteri, ritorni la prima
lettera maiuscola, se c'e'. Se non c'e', allora ritorna '\n'.

Esercizi

6) Scrivere una procedura che ricevuto un array di caratteri, trasformi ogni

lettera minuscola in maiuscola, lasciando inalterati gli altri caratteri

7) Dato un array di interi, scrivere una funzione che controlli che l'array contenga solo elementi pari.

8) Scrivere una funzione che ricevuto un array di interi, ritorni il numero di elementi pari presenti nell'array

9) Scrivere una funzione che ricevuto un array di interi, ritorni la somma degli elementi in posizione pari.

10) Scrivere una funzione che ricevuto un array di interi, ritorni la somma degli
elementi in posizione dispari meno la somma degli elementi in posizione pari.

11) Scrivere una funzione che ricevuto un array di interi, ritorni il massimo elemento dell'array.

12) Dato un array di interi non vuoto, scrivere una funzione che ritorni il numero
di occorrenze del valore minimo (scorrendo l'array una volta sola).

13) Dato un array di interi non vuoto, scrivere una funzione che ritorni il numero
di occorrenze del valore pari minimo (scorrendo l'array una volta sola).

14) Scrivere una procedura che ricevuto un array di interi, lo modifica raddoppiando
ogni elemento.

15) Scrivere una funzione che ricevuto un array di interi, restituisca la posizione
dell'ultima occorrenza di un elemento pari nell'array.
int ultimo_pari(int arr[], int dim);

16) Scrivere una procedura che ricevuto un array di caratteri modifichi il vettore
in modo che ogni vocale venga sostituita dal simbolo '$'.
void dollarize(char arr[], int dim);

17) Scrivere una funzione che verifichi che un vettore sia ordinato in senso descrescente.

18) Scrivere una funzione che riceva una matrice bidimensionale di interi e azzeri
ogni elemento il cui contenuto e' maggiore o uguale alla somma delle sue coordinate.
void azzera(int mat[][5], int numRighe);

19) Dato un array di interi, definire una funzione che controlli che l'array contenga
solo numeri primi.

20) Scrivere una procedura che, ricevuto un array di interi, ordini l'array. Non ci sono restrizioni di efficienza, potete usare un qualsiasi algoritmo di ordinamento (ad esempio la ricerca del minimo nel sotto-array residuo da spostare
nella prima posizione del sotto-array).