8.1 EXTRA: gli array o vettori monodimensionali di caratteri (STRINGHE)
(senza ALGORITMI di esempio)

Gli array o vettori monodimensionali di caratteri (più comunemente conosciute come STRINGHE) sono concettualmente del tutto simili alle medesimo tipo di struttura dati composta da numeri interi, o da numeri reali o da valori booleani.
Pertanto anche per questa tipologia di struttura dati, valgono le medesime considerazioni e le medesime pseudo istruzioni già illustrate nel caso di vettori monodimensionali numerici.

Esempio A) lettura e scrittura di un array monodimensionale di caratteri
(N.B. Come fatto finora per gli array monodimensionali numerici ossia un elemento alla volta)

ALGORITMO LetturaScritturaStringaCaratterexCarattere
MAXDIM 50
PROCEDURA main( )
str: **ARRAY**[MAXDIM] **DI CHAR**
i, n: INT
INIZIO
/ leggo e controllo la dimensione della stringa rispettando il vincolo imposto da MAXDIM */*
RIPETI
  Leggi (n)
FINCHE' (n ≥ 1) **AND** (n ≤ MAXDIM)
/ carico UNO ALLA VOLTA i caratteri nella stringa */*
PER i ← 1 **A** n **ESEGUI**
  Leggi (str[i])
  i ← i + 1
FINE PER
/ visualizza UNO ALLA VOLTA i caratteri precedentemente immessi nella stringa */*
PER i ← 1 **A** n **ESEGUI**
  Scrivi (str[i])
  i ← i + 1
FINE PER

FINE

Iniziamo introducendo due nuove **PSEUDO** funzioni da applicare **ESCLUSIVAMENTE** a **variabili di tipo CHAR**:

x, y : **CHAR**

y ← **Minuscola**(x) restituisce il carattere in minuscolo corrispondente a x oppure lo lascia inalterato

y ← **Maiuscola**(x) restituisce il carattere in maiuscolo corrispondente a x oppure lo lascia inalterato

N.B. E' possibile (ed apparirà da subito estremamente conveniente) introdurre nuove funzionalità nella pseudocodifica dedicate ESCLUSIVAMENTE agli ARRAY MONODIMENSIONALI DI CARATTERI (ossia alle STRINGHE) per ampliare la possibilità di effettuare confronti, copie, assegnazioni oppure conoscere il numero complessivo di elementi, utilizzando l'intera stringa (evitando quindi di dover procedere carattere per carattere)

Particolarità della PSEUDOCODIFICA da applicare alle STRINGHE

Supponendo di avere a disposizione due stringhe ossia due array monodimensionali di caratteri **str**, **str1**: **ARRAY[MAXDIM] DI CHAR** sarà possibile d'ora in poi:

(1)
/* Leggere una stringa per intero */
Leggi (str) //Equivale a leggere **str** un carattere alla volta risparmiando il ciclo

(2)
/* Scrivere una stringa per intero */
Scrivi (str) //Equivale a scrivere **str** un carattere alla volta risparmiando il ciclo

(3)
/* Assegnare ad una stringa un valore iniziale */
str ← **"Pippo"** //Equivale ad assegnare alla stringa **str** un carattere alla volta di "Pippo"
str ← **""** //Equivale ad assegnare alla stringa **str** la cosiddetta STRINGA VUOTA

(4)
/* Assegnare ad una stringa il valore di un'altra stringa */
str ← **"Pippo"** //Equivale ad assegnare alla stringa **str** un carattere alla volta di "Pippo"
str1 ← str //Anche la stringa **str1**, dopo l'assegnazione, avrà valore "Pippo"

(5)
/* Effettuare confronti tra due stringhe */
*Esempio:supponiamo che **str** ed **str1** siano due stringhe opportunamente valorizzate. Potrò scrivere*

SE (str = str1)
ALLORA
<B1>
ALTRIMENTI
<B2>
FINE SE

N.B. ma anche (str ≠ str1) oppure (str < str1) oppure (str ≤ str1) oppure (str > str1) oppure (str ≥ str1

*Il valore dell'enunciato semplice booleano proposto risulterà **VERO** se il confronto tra le due stringhe, effettuato carattere per carattere, darà esito positivo ossia se le due stringhe risulteranno possedere caratteri uguali nelle medesime posizioni.*
(N.B. Si ricorda che tale confronto è CASE-SENSITIVE)

*Il valore dell'enunciato semplice booleano proposto risulterà **FALSO** se il confronto tra le due stringhe, effettuato carattere per carattere, darà esito negativo. Esso verrà ovviamente interrotto al primo carattere delle due stringhe che risulterà diverso nella posizione che si sta considerando.*

(6)
/* Conoscere la lunghezza (ossia il numero di caratteri) di una stringa qualsiasi */
*Esempio:supponiamo che **str** sia una stringa ricevuta in input. Potrò utilizzare la pseudofunzione **Lunghezza( )** che riceve come parametro di ingresso la stringa di cui si vuole conoscere il numero di caratteri e lo restituisce (nel suo nome) come numero intero*
str : **ARRAY[**MAXDIM**] DI CHAR**
n: **INT**
…..
Leggi (str)
n ← **Lunghezza** (str) //N.B. Lunghezza ("") = 0

N.B. Tale valore risulta utilissimo nel caso dei controlli da effettuare su di una stringa che si suppone obbligatoriamente valorizzata (ossia diversa dalla stringa vuota "") e che non superi il massimo numero di caratteri consentito (espresso dalla costante MAXDIM)

(7)
/* effettuare la concatenazione di 2 o più stringhe */

Supponendo di avere a disposizione due stringhe qualsiasi ossia due array monodimensionali di caratteri

str1, **str2**: **ARRAY[MAXDIM] DI CHAR**

ed una ulteriore stringa di caratteri di dimensione pari alla somma delle dimensioni delle due stringhe str1 e str2

str3 : **ARRAY [2* MAXDIM] DI CHAR**

sarà possibile d'ora in poi scrivere:

Leggi (str1)
Leggi (str2)
str3 ← str1 . str2 /*Le due stringhe vengono scritte una dopo l'altra ed il valore assegnato a str3*/

Esempio:
Se la stringa ***str1*** *fosse* ***"Oggi mi sento"*** *e la stringa* ***str2*** *fosse* ***"in formissima!"***
allora l'operazione di concatenazione

str3 ← str1 . str2

farà assumere alla stringa ***str3*** *il valore* ***"Oggi mi sentoin formissima!"***
(Nota Bene senza spazio separatore)

Nel caso volessimo anche la separazione tra le due stringhe basterà prevedere nella concatenazione anche la stringa costituita da un solo spazio "♭" ossia

str3 ← str1 . "♭" . str2

In questo caso la sringa ***str3*** *assumerà il valore* ***"Oggi mi sento in formissima!"***

Diamo ora una versione alternativa dell'utilizzo degli array monodimensionali di caratteri (stringhe) all'interno di un algoritmo utilizzando le particolarità della pseudocodifica appena illustrate

Esempio B) lettura e scrittura di un array monodimensionale di caratteri
(N.B. Utilizzando le facilitazioni introdotte per le stringhe)

```
ALGORITMO LetturaScritturaStringa1
MAXDIM 50
PROCEDURA main( )
str : ARRAY[MAXDIM] DI CHAR
i, n: INT
INIZIO
/* leggo e controllo la stringa TUTTA IN UNA VOLTA rispettando il vincolo imposto da MAXDIM */
RIPETI
  Leggi (str)
  n ← Lunghezza (str)
FINCHE' (n ≥ 1) AND (n ≤ MAXDIM)
/* visualizzo la stringa TUTTA IN UNA VOLTA*/
Scrivi (str)
FINE
```

oppure

```
RIPETI
  Leggi (str)
  n ← Lunghezza (str)
FINCHE'(str ≠ "") AND (n ≤ MAXDIM)
```

N.B. Poichè il nome della pseudofunzione Lunghezza( ) può essere usato (come spiegheremo dettagliatamente in seguito) direttamente come se fosse una variabile ed il valore da essa ritornato essere inserito in un'operazione di controllo, sarà possibile d'ora in poi scrivere:

```
ALGORITMO LetturaScritturaStringa2
MAXDIM 10
PROCEDURA main( )
str: ARRAY[MAXDIM] DI CHAR
i: INT
INIZIO
/* leggo e controllo la stringa TUTTA IN UNA VOLTA rispettando il vincolo imposto da MAXDIM */
RIPETI
  Leggi (str)
FINCHE' (Lunghezza (str) ≥ 1) AND (Lunghezza (str) ≤ MAXDIM)
/* visualizzo la stringa TUTTA IN UNA VOLTA*/
Scrivi (str)
FINE
```

oppure

```
RIPETI
  Leggi (str)
FINCHE'(str ≠ "") AND (Lunghezza (str) ≤ MAXDIM)
```

Particolarità della CODIFICA C da applicare alle STRINGHE

Mettiamo a confronto quanto visto finora sugli array monodimensionali di caratteri (stringhe) nella PSEUDODIFICA (struttura dati astratta) rispetto al linguaggio di programmazione di alto livello C (strutture dati previste dalla sintassi del linguaggio di programmazione scelto).

PREMESSA: Nel linguaggio C le stringhe sono array monodimensionali di caratteri che devono obbligatoriamente prevedere un carattere speciale che indichi la loro fine.
Tale carattere è chiamato CARATTERE DI TERMINAZIONE DELLA STRINGA (o TAPPO) e corrisponde al carattere '\0'.

Inoltre, come nella pseudocodifica, anche in questo caso la chiamata alle funzioni che utilizzeremo per le stringhe (contenute nella libreria di sistema string.h che deve essere inclusa, ad inizio programma, nel caso di loro utilizzo) restituiscono un valore nel nome della funzione stessa che quindi può essere utilizzato direttamente in un'espressione o in un confronto come se di fatto fosse una variabile.

Esempio 1) lettura e scrittura di un array monodimensionale di caratteri – LINGUAGGIO C

```
#include <stdio.h>
#include <stdlib.h>
#define MAXDIM 50
int main (int argc, char* argv[ ])
{
char str [MAXDIM + 1];        // Devo tenere conto del carattere di terminazione della stringa '/0'
int i, n;
/* leggo e controllo la stringa TUTTA IN UNA VOLTA rispettando il vincolo imposto da MAXDIM */
do
  {
  printf("Inserisci la lunghezza della stringa: ");
  fflush(stdin);              //la funzione fflush(stdin) pulisce il buffer dello datndard input (tastiera)
  scanf("%d", &n);
  }
while ((n < 1) || (n > MAXDIM));
/* carico uno alla volta i caratteri nella stringa */
for (i = 0; i < n; i++)
  {
  printf("Immetti il %d^ carattere della stringa: ", i + 1);
  fflush(stdin);
  scanf ("%c", &str[i]);          //alternative str[i] = getc (stdin);  oppure  str[i] = getchar( );
  }
/* N.B. IMPORTANTISSIMO: devo impostare il CARATTERE DI TERMINAZIONE */
str[n] ← '\0';
/* visualizzo uno alla volta i caratteri precedentemente immessi nella stringa */
printf("la stringa immessa e' : ");
for (i = 0; i < n; i++)
  {
  printf ("%c", str[i]);          //alternative  putc (str[i], stdout);  oppure  putchar (str[i]);
  }
system ("PAUSE");
return 0;
}
```

Esempio 2) lettura e scrittura di un array monodimensionale di caratteri – LINGUAGGIO C (ALTERNATIVO SPECIALE)

!!!!!!! ATTENZIONE !!!!!!
La funzione **scanf** applicata alle stringhe (quindi con formato "**%s**") non è in grado di acquisire correttamente una stringa contenente gli spazi.
In caso di stringa in input con uno o più spazi, detta funzione, contrariamente alla **gets** che non da alcun problema, si limiterebbe a restituire tutti i caratteri letti fino al primo spazio (blank) incontrato.
(esempio: **"La rosa dei venti"** letta con **scanf** restituitrebbe solamente **"La"**)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXDIM 50
int main (int argc, char* argv[ ])
{
char str [MAXDIM + 1];          // Devo tenere conto del carattere di terminazione della stringa '/0'
int i, n;

/* leggo e controllo la stringa TUTTA IN UNA VOLTA rispettando il vincolo imposto da MAXDIM */
do
{
  printf("Inserisci la stringa: ");
  fflush(stdin);
  scanf ("%s", str);        //alternativa gets (str);
  n = strlen (str);                                  oppure
}
while ((n = = 0) || (n > MAXDIM));

/* visualizzo la stringa TUTTA IN UNA VOLTA */
printf("La stringa immessa e': ");
printf ("%s", str);          //alternativa puts (str);

system ("PAUSE");
return 0;
}
```

```
do
  {
  printf("Inserisci la stringa: ");
  fflush(stdin);
  gets (str);
  n = strlen (str);
  cfr = strcmp (str, "");
  }
while ((cfr = = 0) || (n > MAXDIM));
```

N.B. Traducendo fedelmente il nostro pseudocodice avremmo dovuto scrivere
while ((n < 1) || (n > MAXDIM));
ma poiché non esistono stringhe a lunghezza negativa e la lunghezza della stringa vuota "" è ovviamente uguale a 0, possiamo scrivere anche così!

OPPURE IN ALTERNATIVA

```
/* leggo e controllo la stringa TUTTA IN UNA VOLTA rispettando il vincolo imposto da MAXDIM */
do
{
  printf("Inserisci la stringa: ");
  fflush(stdin);                                  oppure
  scanf ("%s", str);   //alternativa gets (str);
}
while ((strlen(str) = = 0) || (strlen(str) > MAXDIM));

/* visualizzo la stringa TUTTA IN UNA VOLTA */
printf("La stringa immessa e': ");
printf ("%s", str);    //alternativa puts (str);

system ("PAUSE");
return 0;
}
```

```
do
  {
  printf("Inserisci la stringa: ");
  fflush(stdin);
  gets (str);
  }
while ((strcmp (str, "") = = 0) || (strlen (str) > MAXDIM));
```

Breve riassunto delle funzioni del linguaggio C fin qui utilizzate

a) convertire in minuscolo/maiuscolo un carattere: nell'ipotesi che sia stata definita la seguente variabile:

```
char x;
```

- x = **tolower** (x); //conversione del carattere x in minuscolo
- x = **toupper** (x); //conversione del caratterex in MAIUSCOLO

b) leggere un singolo carattere: nell'ipotesi che sia stata definita la seguente variabile:

```
char x;
```

- **scanf** ("%c", **&**x);
- x = **getc** (stdin);
- x = **getchar** ( );

c) scrivere un singolo carattere: nell'ipotesi che sia stata definita la seguente variabile:

```
char x;
```

- **printf** ("%c", x);
- **putc** (x, stdout);
- **putchar** (x);

d) leggere una stringa: nell'ipotesi che sia stata definita la seguente variabile:

```
char str[MAXDIM + 1];
```

- **scanf** ("%s", str); //N.B. la lettura si interrompe al primo blank (spazio) incontrato
- **gets** (str);

e) scrivere una stringa: nell'ipotesi che sia stata definita la seguente variabile:

```
char str[MAXDIM + 1];
```

- **printf** ("%s", str); //N.B. la scrittura NON si interrompe al primo blank (spazio) incontrato
- **puts** (str);

f) calcolare la lunghezza di una stringa: nell'ipotesi che siano state definite la seguenti variabili:

```
char str[MAXDIM + 1];
int n;
```

- n = **strlen** (str);

N.B. Si può usare direttamente la chiamata strlen(str) in una espressione o in un confronto senza dover salvare il valore di ritorno in una apposita variabile intera.
Il CARATTERE DI TERMINAZIONE NON VERRA' OVVIAMENTE INCLUSO NEL CALCOLO

g) copiare una stringa in un altra: nell'ipotesi che siano state definite le seguenti variabili:

```
char str[MAXDIM + 1], str1[MAXDIM + 1];
```

- **strcpy** (str1, str); *//Copia il valore della stringa **origine str** nella stringa di **destinazione str1***

*N.B. Affinchè la funzione esegua in maniera corretta ciò che si prefigge, appare del tutto evidente che la lunghezza della stringa di destinazione **str1** debba essere maggiore o uguale alla lunghezza della stringa origine **str***

h) effettuare un confronto tra due stringhe: nell'ipotesi che siano state definite le seguenti variabili:

```
char str[MAXDIM + 1], str1[MAXDIM + 1];
int cfr;
```

- cfr = **strcmp** (str, str1);

*N.B. Confronta carattere per carattere la stringa **str** con la stringa **str1** seguendo la codifica ASCII restituendo:*

- ***il valore 0*** *se le stringhe str ed str1 coincidono (carattere per carattere)*
- ***un valore < 0*** *se la stringa str ha un carattere con codifica ASCII minore di quello in posizione omologa contenuto nella stringa str1*
- ***un valore > 0*** *se la stringa str ha un carattere con codifica ASCII maggiore di quello in posizione omologa contenuto nella stringa str1*

Si può usare, come abbiamo già visto, direttamente la chiamata strcmp (str, str1) in un confronto senza dover salvare il valore di ritorno in una apposita variabile intera.

i) effettuare la concatenazione tra due (o più) stringhe:
nell'ipotesi che siano state definite le seguenti stringhe (array di char)

```
char str1[MAXDIM + 1], str2[MAXDIM + 1];
char str3[2*MAXDIM +  1];
```

allora

```
gets (str1);              //Leggo la stringa str1
gets (str2);              //Leggo la stringa str2
strcpy (str3, str1);      //Copio la stringa str1 in str3
strcat (str3, str2);      //Concateno la stringa str2 a str1
puts (str3);              //Scrivo la stringa ottenuta dalla concatenazione di str1 e str2
```

nel caso volessimo anche separare le stringhe str1 e str2 con uno spazio allora

```
gets (str1);              //Leggo la stringa str1
gets (str2);              //Leggo la stringa str2
strcpy (str3, str1);      //Copio la stringa str1 in str3
strcat (str3, " ");       //Concateno la stringa stringa formata da uno spazio a str1
strcat (str3, str2);      //Concateno la stringa str2 alla stringa formata da str1 più uno spazio
puts (str3);              //Scrivo la stringa ottenuta dalla concatenazione complessiva
```