

Insegnamento di Programmazione (6 CFU)
Anno accademico 2005/2006
Corso di Laurea in Ingegneria Energetica

Esercizi Parte 5
(stringhe)

Prof. Nadia Ranaldo

Esercizio 1

Scrivere una funzione che riceva una stringa che rappresenta una parola in inglese al singolare e la modifica in modo da ottenere la parola al plurale.

Utilizzare le seguenti regole grammaticali inglesi:

- Se la parola finisce per *s*, *x*, *z*, *ch* o *sh*, aggiungere *es* alla parola.
- Se la parola finisce per *y* e la *y* è preceduta da una consonante, cambiare *y* in *ies*.
- In tutti gli altri casi, basta aggiungere *s* alla parola.

Esercizio 2

Scrivere una funzione predicato

```
int equalIgnoringCase(char *s1, char *s2)
```

che restituisce **TRUE** se le stringhe **s1** e **s2** sono uguali, ignorando le differenze di lettere maiuscole/minuscole. Per esempio, **EqualIgnoringCase("CAT", "cat")** deve restituire **TRUE**.

Esercizio 3

Il concetto di palindromo si riferisce a parole o frasi che si leggono indifferentemente da destra a sinistra e viceversa.

- a) Scrivere una funzione predicato **IsPalindrome(str)** che restituisce **TRUE** se la stringa **str** è una parola palindroma e provarla all'interno di un programma.
- b) Nel caso di frasi palindrome si ignorano tutti i segni di punteggiatura e le differenze tra lettere maiuscole e minuscole. Per esempio, la frase

Madam, I'm Adam.

è una frase palindroma, in base alle regole descritte. Scrivere una funzione predicato **IsSentencePalindrome(str)** che restituisce **TRUE** se la stringa **str** rispetta la definizione di frase palindroma. Per esempio, si dovrebbe scrivere un programma in grado di usare la funzione per ottenere la seguente uscita sullo schermo:

Questo programma verifica le frasi palindrome.

Indicare la fine dell'input con un INVIO.

Immettere una stringa: Madam, I'm Adam.

Questa frase è palindroma.

Immettere una stringa: A man, a plan, a canal: Panama!

Questa frase è palindroma.
Immettere una stringa: Non sono una frase palindroma.
Questa frase non è palindroma.
Immettere una stringa:.

Esercizio 4

Scrivere la funzione predicato

```
int isSubString(char *str, char *sub)
```

che determina se la stringa **sub** è contenuta nella stringa **str**.
Ad esempio “cia” è contenuta in “ciao mondo”, mentre “cio” non è contenuta.

Esercizio 5

Scrivere una funzione predicato

```
int isUpperCase(char *str)
```

che determina se la stringa contiene tutte lettere maiuscole.
Ad esempio passando come parametro “CIAO” la funzione deve restituire vero, passando come parametro “Ciao” la funzione deve restituire falso.

Esercizio 6

Scrivere una funzione

```
void concat(char *str1, char *str2, char *str3)
```

che concatena tre stringhe passate come parametro. Il risultato deve essere inserito nella prima stringa (**str1**).

Ad esempio **str1="cia" str2="o mo" str3="ndo"**

Al termine della funzione **str1="ciao mondo"**.

Si supponga che la stringa **str1** contenga spazio sufficiente per memorizzare i caratteri di tutte e tre le stringhe.

Scrivere una versione che utilizza la funzione di libreria **strcat** e un'altra che non utilizza tale funzione.

Esercizio 7

Scrivere una funzione

```
void concat1(char *str1, char *str2, char *str3)
```

che concatena tre stringhe passate come parametro. Il risultato deve essere inserito nella prima stringa (**str1**).

Ad esempio **str1="cia" str2="o mo" str3="ndo"**

Al termine della funzione **str1="ciao mondo"**.

Se la stringa `str1` non ha spazio sufficiente per contenere tutti i caratteri delle tre stringhe, la funzione deve troncarsi i caratteri che non possono essere memorizzati (ricordando di dover sempre inserire lo `'\0'` alla fine).

Ad esempio se `str1` può contenere 9 caratteri e
`str1="cia" str2="o mo" str3="ndo"`

al termine della funzione `str1="ciao mon"` (8 caratteri più lo `'\0'` alla fine).

Scrivere una versione che utilizza la funzione di libreria `strcat` e un'altra che non utilizza tale funzione.

Esercizio 8

Implementare la funzione `isConsonant(char ch)` che restituisce `TRUE` se `ch` è una consonante. La funzione dovrebbe essere in grado di riconoscere sia lettere maiuscole che minuscole. Scrivere un programma che mostri tutte le consonanti maiuscole dell'alfabeto utilizzando un ciclo su tutte le lettere maiuscole.

Esercizio 9

Scrivere una funzione `randomWord` che restituisce una parola (stringa) costruita mediante una scelta casuale delle lettere. Il numero delle lettere nella parola dovrebbe essere scelto anch'esso casualmente, selezionandolo in un intervallo limitato dalle costanti `MinLetters` e `MaxLetters`, opportunamente definite. Scrivere un programma che utilizzi la funzione `randomWord` per visualizzare 6 parole casuali.

Esercizio 10

Scrivere una funzione

```
void dateString(int day, int month, int year)
```

che restituisca una stringa costituita dal giorno del mese, un trattino, le prime tre lettere del nome del mese, un altro trattino, e le ultime due cifre dell'anno. Per esempio, invocando la funzione

```
dateString (22, 11, 1963)
```

si dovrebbe avere la stringa risultato `"22-Nov-63"`.

Esercizio 11

Scrivere le vostre versioni delle funzioni della libreria `string.h` per la ricerca nelle stringhe `strchr` e `strrchr`.

Prototipi delle funzioni da realizzare:

```
char *strchr1(char *s, int c)
```

Individua la prima occorrenza del carattere `c` nella stringa `s`. Qualora `c` sia stato trovato, sarà restituito un puntatore alla locazione di `c` in `s`. Altrimenti sarà restituito un puntatore `NULL`.

```
char *strrchr1(char *s, int c)
```

Individua l'ultima occorrenza del carattere `c` nella stringa `s`. Qualora `c` sia stato trovato, sarà restituito un puntatore alla locazione di `c` in `s`. Altrimenti sarà restituito un puntatore `NULL`.