Sistemi operativi – laboratorio

Esercitazione 0: elementi di C per sistemi operativi

09/03/2020

Ex_1 [command line arguments]:

Realizzare un programma C che chiede all'utente di digitare il nome del programma stesso. Se l'utente risponde correttamente il programma termina e ringrazia, altrimenti risponde "No, questo non e' il mio nome!" e richiede di nuovo di inserire il nome corretto. Risolvere il problema utilizzando gli argomenti da linea di comando in modo da rendere il programma generale rispetto al nome.

Ex_2 [command line arguments]:

Realizzare un programma C che simula una calcolatrice semplificata. Il programma riceve da *linea di comando*:

- un operatore tra +, -, *, /

- due numeri

e restutuisce il risultati dell'operazione. Per semplicità si può trascurare la gestione degli errori.

Ex_3 [ascii coding]:

Realizzare un programma C che chiede all'utente un numero intero maggiore di zero, lo acquisisce come *int* e ne restituisce la sequenza di caratteri inversa. Ad esempio, dato il numero 156, restituisce 651. Nota: la sequenza di caratteri deve essere stampata per mezzo del comando *putchar(c)*, dove *c* acquisisce di volta in volta il carattere relativo ad una cifra del numero in input. Suggerimento: utilizzare la trasformazione da intero ad ascii (vedi tabella nelle slide). Qual'e' lo shift da aggiungere? Cosa succede se anziche' aggiungere lo shift corretto si aggiunge 65?

Ex_4 [bitwise operators]:

Scrivere una funzione C chiamata *bit_print* che riceve in input un numero intero (int) e ne stampa la rappresentazione macchina (stringa di bit). Implementare poi un metodo *main* che

- stampa i numeri da 0 a 9 in rappresentazione binaria

- definisce gli interi a=33333 e b= -77777, e stampa le stringhe di bit di a, b, a&b, a^b, a|b, ~(a | b), (~a & ~b), a<<1, a<<4, a<<31, a>>4.

Ex_5 [macros]:

Scrivere un programma C che definisce una *macro* (con il comando #define) per il calcolo del quadrato di un numero. Testare la macro con i seguenti input:

- 7

- 7+5

Cosa succede eliminando alcune parentesi (vedi esempio slide)?

Ex_6 [macros]:

Testare la direttiva di compilazione condizionata *ifdef* (vedere documentazione C) generando un file *myHeader.h* che definisce una funzione *myFun()* solo se una costante *M1* e' stata definita a livello preprocessore. Il metodo *main*, che include *myHeader*, utilizza la funzione *myFun()*. Cosa accade se prima del suo utilizzo non si definisce la costante *M1*?

Ex_7 [structure]:

Scrivere una *struttura* di nome *student* con i seguenti membri: *last_name* (stringa), *student_id* (int), *grade* (carattere tra 'A' ed 'F'). Generare poi un metodo *main* che crea un array di 5 studenti e restituisce il numero di voti negativi per mezzo di una funzione *fail* che conta come negativo ogni voto uguale ad 'F' nell'array di studenti. Utilizzare l'operatore *typedef* per dare un nome sintetico alla struttura. Il metodo *main* deve inoltre restituire il cognome del primo studente accedendo al relativo membro per mezzo dell'operatore "." ed il voto del primo studente accedendo al relativo membro per mezzo dell'operatore "->".

Ex_8 [union]:

Scrivere un programma C che definisce una *union* avente due membri, un *int* ed un *float*. Mostrarne l'utilizzo nel caso d'esempio definito nelle slide (sezione "Unions", slide 33).

Ex_9 [pointers]:

Realizzare un programma C che genera una variabile per la stringa "Corso di sistemi operativi" e chiama una funzione *void stringToMultiDimArray(char *s)* che trasforma la stringa in un array bidimensionale (utilizzando un puntatore a puntatore) che contiene in ciascuna posizione una parola della stringa originale. Suggerimento: utilizzare la funzione *strtok* per spaccare la stringa originale in parole.