Esercizi
Programmazione I e Informatica II

28 Ottobre 2015

Esercizio 1 Stampa di una valore puntatore

Definire una variabile con identificatore *a* e tipo *int*, ed un puntatore di nome *pA* a cui assegnare il valore della locazione di memoria dove si trova *a*. Stampare il valore del puntatore utilizzando la formattazione *%p*. Stampare anche il valore $pA + 1$. Di quanto aumenta (in decimale) questo valore rispetto a *pA*? *pA + 1* è un *lvalue*?

Esercizio 2 Valutazione espressioni

Date le seguenti espressioni, raggruppare gli operndi con parentesi tonde in base seguendo l'ordine di valutatazione dell'espressione, e darne il valore finale. *char c = 'w'; int i = 1, j = 2, k = -7; double x=7e+33, y= 0.001.*

- $'a' + 1 < c$
- $-i - 5 * j >= k + 1$
- $3 < j < 5$
- $x - 3.333 <= x + y$
- $x < x + y$

Esercizio 3 Stampa dimensione tipi

Stampare la dimensione in byte dei seguenti tipi: *char*, *short*, *int*, *long*, *unsigned*, *float*, *double*, *long double*. Stampare anche la dimensione di *int a[3]*. Perché se utilizzo il formato *%d* ottengo un warning dal compilatore?

Esercizio 4 Programmazione modulare

Creare una libreria con 8 funzioni. Le definizioni di queste funzioni sono contenute in un file *mylibrary.c*. Esse sono: *add()*, *sub()*, *mult()*, *div*, che prendono come parametri due valori di tipo *int* e restituiscono un valore di tipo *int*, che corrisponde alla semplice applicazione dell'operazione definita dal nome

della funzione sui due parametri. In più, ci sono le versioni più precise di esse: *add_precise()*, *sub_precise()*, *mult_precise()*, *div_precise*. Esse prendono in input due parametri di tipo *double* e restituiscono un valore di tipo *double*.

Creare anche un file con solo le dichiarazioni di queste funzioni, *mylibrary.h*. In questo file si troveranno solo i prototipi delle funzioni (abbimo già visto l'esempio *int abs(int);*).

Includere la dichiarazione dei prototipi in *mylibrary.c* utilizzando la direttiva *include "mylibrary.h"*. Compilare la libreria utilizzando `gcc -c mylibrary.c`. Si otterrà un file *mylibrary.o*. Cercare quale step di *gcc* non viene utilizzato quando si usa il flag *-c* (per farlo, `gcc -help`).

Creare un terzo file *mainfile.c* che utilizza tutte e otto le funzioni su valori a piacere. Salvare il risultato delle quattro funzioni su *double* in quattro variabili, e poi stampare il valore di ciascuna variabile. Per quanto riguarda le funzioni su *int*, chiamarle (sempre con valori a piacere) direttamente dentro *printf()* senza salvare il risultato in una variabile.

Per compilare *mainfile.c*, includere anche il file con i prototipi *mylibrary.h* utilizzando la direttiva *include "mylibrary.h"*. Compilare questo file utilizzando `gcc -c mainfile.c`: può essere compilato senza sapere il codice delle funzioni in *mylibrary.c*.

Infine, per eseguire veramente il programma compilare con `gcc -o mainfile mainfile.o mylibrary.o`, ed eseguire *./mainfile*. Cosa succede compilando in questo modo? Quale parte di `gcc` viene utilizzata adesso?