

Esercitazioni di Fondamenti di Informatica - Lez. 6 29/10/2020

Esercizi sullo *scope* delle variabili e sulle funzioni *ricorsive* in C

I codici e le soluzioni degli esercizi sono nella cartella codice6

Se una *variabile locale* (per esempio interna a una funzione) ha lo stesso nome di un'altra *variabile globale*, **la variabile locale nasconde sempre la variabile globale**. Questo significa che dentro alla funzione verrà sempre usata la variabile locale.

Analizziamo di seguito dei semplici programmi per capire cosa stampano al termine della loro esecuzione.

esempio 1

```
#include<stdio.h>
int a = 1; /* variabile globale */

void f(int a){
    a=2;
}
void g(int *a){
    *a=3;
}
int h(){
    return a; /* restituisce il valore della variabile globale */
}

int main(){

    printf("%d\n",a); /* stampa la variabile globale a=1 */

    int a = 4;
    printf("%d\n",a); /* stampa la variabile locale a=4 */

    f(a);
    printf("%d\n",a); /* stampa la variabile locale a = 4 */
    g(&a);
    printf("%d\n",a); /* stampa a = 3 */
    a= h();
    printf("%d\n",a); /* stampa a = 1 */
}
```

esempio 2

```
// esempio 2 cosa stampa

#include<stdio.h>

int f(int *b){
    static int a = 1; // vale solo la prima chiamata
    *b +=a; // *b= *b+a;
    a +=1; // a= a+1;
    return a;
}

int main(){

    int a = 10;

    f(&a);
    printf("a = %d\n",a); /* stampa a che vale 11 */

    f(&a);
    printf("a = %d\n",a); /* a locale stampa 13*/

    a = f(&a); /* stampa 4 */

    printf("a = %d\n",a);
}
```

esempio 3

```
// esempio 3 cosa stampa

#include<stdio.h>

int f(int n){
    if (n==0){
        return 1;
    }
    else{
        return n * f(n-1);
    }
}

int main(){

    int b = f(6);

    printf("%d\n", f(6) );
}
```

NOTA esempio3

Ad ogni passo ricorsivo , il numero dato in input viene moltiplicato per il suo predecessore.

Dato che la ricorsione termina quando l'input è 1 o 0, si ottiene il seguente prodotto: $n * (n-1) * (n-2) * (n-3) \dots * 2 * 1$

→ il risultato e' il fattoriale del numero 6

1. Scrivere una funzione ricorsiva che calcoli il massimo di un array di interi. La funzione deve ricevere in input l'array e la sua dimensione.

Soluzione:

```
#include<stdio.h>
int massimo(int *array, int n){

    if(n==0){
        return 0;
    }
    else if(n==1){
        return *array;
    }
    else{
        int max = massimo(array +1, n-1);

        if(array[0]> max){
            return array[0];
        }
        else{
            return max;
        }
    }
}
int main(){

    int i[]={2,6,55,7,2};
    printf("stampo max %d \n",massimo(i,5));
}
```

2. Scrivere una funzione ricorsiva che calcoli la lunghezza di una frase terminata da '\0'.

Soluzione:

```
#include<stdio.h>

int lunghezza(char * frase){

    /* caso base: l'array e' vuoto: */
    if(*frase == '\0'){
        return 0;
    }
    else{
        return lunghezza(frase +1) +1;
    }
}

int main(){
    char frase[] = "Q\0";
    printf("Lunghezza frase: %d\n",lunghezza(frase));
}
```

3. Scrivere una funzione ricorsiva che trasformi in maiuscole le lettere minuscole di una frase. La funzione riceve un array di caratteri terminato da '\0'.

Soluzione:

```
#include<stdio.h>
#define MAX 100

void maiuscolo(char *frase){

    if(*frase!='\0'){

        /* se il carattere e' una lettera minuscola */
        if(*frase>='a' && *frase <='z'){

            *frase -=('a' - 'A'); // *frase =*frase - ('a' - 'A')
        }
        maiuscolo(frase +1);    } }

}
```

```
int main(){  
    char frase[MAX] = "Questa e' una frase\0";  
    printf("La frase \n%s \n e' trasformata in: \n",frase);  
    maiuscolo(frase);  
    printf("%s\n",frase);  
}
```

4. Scrivere una funzione ricorsiva che dati un array di interi e la sua dimensione, restituisca la somma dei prodotti tra il primo elemento e l'ultimo, il secondo e il penultimo,...

Es: $a = \{1, 2, 3, 4, 5\} \rightarrow 1 \cdot 5 + 2 \cdot 4 + 3 \cdot 3$

Soluzione:

```
#include<stdio.h>  
  
int f(int *array, int n){  
    if(n<=0){  
        /* caso base: se l'array ha dimensione 0 o inferiore a 0 */  
        /* nota: se usassimo solo n==0 come condizione la funzione non  
        terminerebbe mai con array di dimensione dispari */  
        return 0;  
    }  
    else if (n==1){  
        return array[0]*array[0];  
    }  
    else{  
        /* passo ricorsivo: moltiplichiamo il primo e l'ultimo elemento  
        dell'array, chiamando poi in modo ricorsivo la funzione sull 'array stesso  
        meno il primo e  
        l'ultimo elemento */  
        return array[0] * array[n-1] + f(array +1, n-2);  
    }  
}  
  
int main(){  
    int array[] = {1,2,3,4,5};  
    printf("Il risultato e' %d\n",f(array,5));  
}
```

5. Scrivere la funzione ricorsiva che dati: un array di 1 e 0 che rappresenta un numero in base 2 e la sua dimensione, effettui la conversione da base 2 a base 10.

Soluzione con funzione ausiliaria:

```
#include<stdio.h>
/* versione con funzione converti */

int converti(int *array, int n, int potenza){

    if(n==0){
        /* caso base: array di dimensione 0*/
        return 0;
    }
    else{
        return array[n-1] * potenza + converti(array, n-1, potenza*2);
    }
    /* chiamata ricorsiva: calcolo il valore del bit meno
    significativo e lo aggiungo al valore della conversione del restante
    array */
}

int main(){

    int array[] = {1,0,1};

    printf("Risultato %d\n",converti(array, 3, 1));

}
```

Soluzione con funzione potenza:

```
#include<stdio.h>
/* versione con funzione potenza */
/* funzione ausiliaria per il calcolo della potenza */

int potenza(int base, int esponente){

    if(esponente ==0){
        return 1;
    }
    else{
        return base * potenza(base,esponente-1);
    }
}

int converti(int * array, int n){
```

```

    if(n==0){                /* caso base: array di dimensione 0*/
        return 0;
    }

    else{
        return array[0]*potenza(2,n-1) + converti(array+1,n-1);
    }

/* calcolo esplicitamente la potenza */

/* chiamata ricorsiva: calcolo il valore del bit meno significativo e lo
aggiungo al valore della conversione del restante array */
}

int main(){
    int array[] = {1,0,1};
    printf("Risultato %d\n",converti(array,3));
}

```

Soluzione con variabile static:

```

#include<stdio.h>
/* approccio simile all'esercizio es08_a:

    invece di calcolare la potenza esplicitamente con un funzione
    ausiliaria , la calcolo ad ogni passo ricorsivo.
    Il suo valore viene memorizzato all'interno di una variabile statica.
    */

int converti(int * array, int n){
/* valore iniziale potenza bit meno significativo */
    static int potenza = 1;

    if(n==0){
        potenza = 1;
        return 0;
    }

/* al termine del calcolo riporto il valore della potenza al valore
originale per evitare errori

/* caso base: array di dimensione 0*/
    }
    else{

        int p = array[n-1] *potenza;
        potenza *=2;
/* valore decimale bit meno significativo */

        return p + converti(array, n-1);
    }

/* chiamata ricorsiva: il valore del bit meno significativo e' aggiunto
al valore della conversione del restante array */
}

int main(){

```

```
    int array[] = {1,0,1};  
    printf("Risultato %d\n",converti(array,3));  
}
```

6. Creare delle funzioni che ricorsivamente stampino un array di strutture persona.

```
typedef struct{  
    char nome[MAX];  
    char cognome[MAX];  
}Persona;
```

Soluzione:

```
#include<stdio.h>  
#define MAX 30  
  
/* definizione della struttura Persona */  
typedef struct{  
    char nome[MAX];  
    char cognome[MAX];  
}Persona;  
  
void stampa_persona(Persona * persona){  
    printf("%s %s\n", persona->nome,persona->cognome);  
}  
  
void stampa_personi(Persona * persone, int n){  
    /* N.B. se non ci sono persone , allora termina */  
    if(n>0){  
        // stampa_persona(persone); /* stampa dati di una persona */  
        stampa_personi(persone+1,n-1);  
  
        /* chiamata ricorsiva per stampare le persone rimanenti */  
  
        /* nota: se la stampa viene fatta prima della chiamata ricorsiva ,  
        allora le persone saranno stampate nello  
        stesso ordine in cui sono memorizzate nell'array,altrimenti in ordine  
        inverso */  
    }  
}  
  
int main(){  
    /* dichiarazione di un array di strutture */
```



```
Persona persone[] = { {"Mario", "Rossi"},  
                      {"Filippo", "Bianchi"},  
                      {"Giulia", "Verdi"}  
};  
  
stampa_persono(persone,3);  
}
```

7. Date le strutture Libro e Libreria, scrivere una funzione ricorsiva che stampa i libri (se presenti) e la giacenza

```
typedef struct{  
    char nome[STRLEN];  
    double costo;  
    int giacenza;  
}Libro;  
  
typedef struct {  
    Libro libri[NLIBRI];  
    int numero_libri;  
}Libreria;
```

// esempio libreria={{“t1”,20,3};{“t2”,20,4};{“t5”,20,1},3}

Soluzione:

```
#include<stdio.h>  
#define STRLEN 30  
#define NLIBRI 100  
  
typedef struct{  
    char nome[STRLEN];  
    double costo;  
    int giacenza;  
}Libro;  
  
typedef struct {  
    Libro libri[NLIBRI];  
    int numero_libri;  
}Libreria;  
  
void stampa_libri(Libreria libreria){  
    /* se ci sono libri in libreria:*/  
    if(libreria.numero_libri>0){  
  
        Libro libro = libreria.libri[libreria.numero_libri-1];  
  
        if(libro.giacenza>0){  
            printf("%s %d\n",libro.nome, libro.giacenza);  
            libreria.numero_libri-;  
  
            /* creo una copia del libro corrente per comodita ' e  
            semplificare la lettura del codice */  
  
            /* se il libro e' presente in libreria stampo i suoi dati */  
            /* decremento il numero di libri ancora da stampare
```

N.B. -> questa operazione non ha effetto sulla struttura originale ,
perche ' fatta su di una copia */

```
        stampa_libri(libreria);    /* passo ricorsivo */
    }
}
int main(){
/* Inizializzazione libreria */
    Libreria libreria = {{
        {"Informatica",10.0,3},
        {"Matematica",20.0,5},
        {"Statistica",5.0,2}
    },3};

    stampa_libri(libreria);    }
```

8. Cosa fanno le funzioni f e g e cosa stampa il programma?

```
void f(int a){
    if(a>0){
        f(a/2);
        printf("%d", (a%2));
    }
}
void g(int a){
    if(a>0){
        printf("%d", (a%2));
        g(a/2);
    }
}
int main(){
    int a;
    scanf("%d",&a);
    f(a);
    printf("\n");
    g(a);
    printf("\n");
}
```

Soluzione:

Entrambe le funzioni f() e g() calcolano e stampano la conversione binaria di un numero decimale.

Quello che cambia tra le due è l'ordine della stampa, che nella funzione f() viene fatta dopo la chiamata ricorsiva.

In questo modo, f() stampa il numero binario dal bit più significativo a quello meno, mentre g() fa il contrario.