

Fondamenti di Informatica e Laboratorio T-AB
Ingegneria Elettronica e Telecomunicazioni

Lab 13

Funzioni ricorsive

Esercizio 1

Scrivere una funzione ricorsiva:

```
int ric(int x)
```

che calcoli, ricorsivamente, la somma di tutti i numeri compresi tra 0 ed x.

Esercizio 1 – Soluzione

```
int ric(int x) {  
    if (x == 0)  
        return 0;  
    else  
        return x + ric(x-1);  
}
```

Esercizio 2

Scrivere una procedura ricorsiva:

void print(int list[], int length)

che stampi, ricorsivamente, tutti i numeri contenuti nell'array **list**.

Esercizio 2 – Soluzione

```
void print (int list[], int length) {
    if (length != 0)
    {
        print(list, length-1);
        printf("%d\n", list[length-1]);
    }
}
```

Esercizio 3

Scrivere una funzione ricorsiva che, ricevuto in ingresso un array di interi, esegua la somma degli interi in posizione con indice dispari.

Esercizio 3 – Soluzione

```
int sumOdd2(int list[], int length, int pos) {
    if (pos >=length)
        return 0;
    else
        return list[pos] + sumOdd2(list, length, pos+2);
}

int sumOdd(int list[], int length) {
    return sumOdd2(list, length, 1);
}
```

Esercizio 3 – Soluzione

(variante)

```
int sumOdd(int list[], int length) {
    if (length >=0) {
        if ( length>=0 && ((length-1)%2) == 1 )
            return sumOdd(list, length-1);
        else
            return list[length-1] + sumOdd(list, length-2);
    }
    else
        return 0;
}
```

Esercizio 4

Si scrivano le versioni ricorsiva ed iterativa (utilizzo di `while`) di una funzione:

`double f(double a, int n);`

che calcoli il seguente valore:

$$\sum_{i=1}^n \left(a - \frac{i}{a} \right)$$

Esercizio 4 – Soluzione

```
double f(double a, int n)
{ if (n==1) return a - 1/a;
  else return a - n/a + f(a, n-1);
}
```

```
double f(double a, int n)
{ int i=1;
  double sum=0;
  while(i<=n)
    {sum = sum + a - i/a;
     i++;}
  return sum;
}
```

Esercizio 5

Si definisca una procedura ricorsiva:

```
void somme2(int l1[], int length)
```

che, ricevuto in ingresso un array l1 di interi, stampi a video gli interi dell'array di ingresso il cui valore è uguale alla somma dei due interi seguenti nell'array (a tal fine, gli ultimi due numeri di un'array sono automaticamente esclusi). Ad esempio, se invocata con l1 = [5, 6, 4, 2, 1, 1, 3, 1], la procedura deve stampare [6, 2]. Infatti, considerando il primo valore (5): i due valori successivi sono 6 e 4, e la loro somma vale 10; quindi 5 è scartato. Per il secondo valore, pari a 6, la somma dei due valori successivi è proprio 6 e quindi il valore viene selezionato per la stampa.

Esercizio 5 – Soluzione

```
void somme2(int l1[], int length) {
    if (length<3)
        return;
    else {
        if (l1[0] == l1[1] + l1[2]) {
            printf("%d, ", l1[0]);
        }
        somme2(&(l1[1]), length-1);
    }
}
```

Esercizio 6

Si scriva un programma che inverta le cifre di un numero intero N usando una funzione apposita. A tal fine, si realizzi sia una versione ricorsiva, sia una versione iterativa della funzione.

Per esempio:

dato $N=4325$, il programma stampa: 5234

Esercizio 6 – Soluzione

```
int reverse2(int num, int part) {
    if (num == 0)
        return part;
    else {
        return reverse2(num/10, part*10 + num%10);
    }
}
```

```
int reverse_it(int num) {
    int result = 0;
    while (num!=0) {
        result = result*10 + num%10;
        num = num/10;
    }
    return result;
}
```

Esercizio 7

In un file `tratte.txt` sono memorizzate per riga tutte le tratte di volo mondiali. Ciascuna riga contiene diversi dati separati dal carattere `,`. In particolare in terza e quinta posizione vi sono rispettivamente i codici degli aeroporti di partenza e arrivo (codici di 3 lettere).

Es: `A3, 96, FCO, 1555, ATH, 3941, , 0, 320`

Scrivere un programma che legga il file, e memorizzi in un array di opportune strutture le tratte.

Esercizio 7

In seguito si definisca la funzione 'cerca' che dati 2 codici di aeroporto stampi a video un possibile percorso di massimo 4 voli che le collegano.

La funzione deve verificare se esiste un volo da-a ed in caso positivo stamparlo e restituire 1.

In caso negativo deve cercare tutti i voli che arrivano in a (voli scalo-a) e cercare soluzioni da-scalo attraverso una opportuna chiamata **ricorsiva**. Se la ricerca ha successo la funzione stampa la tratta scalo-a.

Se il numero di aeroporti visitati è maggiore di 4 la funzione restituisce 0;

Esercizio 7

```
int cerca(char* da, char* a, Tratta* tratte,  
         int nTratte, char* visitati[], int  
         nVisitati)
```

`da`: codice aeroporto partenza

`a`: codice aeroporto arrivo

`trate`: array di tutte le tratte

`nTratte`: numero di elementi nell'array `trate`

`visitati`: array di stringhe. Ciascuna è il codice di un aeroporto già visitato.

`nVisitati`: numero di elementi nell'array `visitati`

Valore di ritorno: 1 in caso di successo, 0 altrimenti.

Esercizio 7

Nel main aggiungere quanto occorre per chiedere all'utente i codici di 2 aeroporti e cercare le soluzioni attraverso la funzione cerca.

In fine provare a definire una funzione cercatutti con logica ricorsiva TAIL che produca a video tutte le possibili soluzioni da A a B in massimo 4 passi

Una possibile signature è la seguente

```
void cercatutti(char* da, char* a, Tratta* tratte, int nTratte,  
char* visitati[], int nVisitati, int* trovati, int nTrovati)
```