E-06: FUNZIONI E VETTORI C

FONDAMENTI DI INFORMATICA E LABORATORIO T-AB

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

UNIVERSITÀ DI BOLOGNA, A.A. 2008/2009

Paolo Torroni, Marco Montali

6 Aprile 2009

Esercizio 1.

Si vuole realizzare un programma che, data da input una sequenza di N parole (ognuna, al massimo, di 20 caratteri), stampi in ordine inverso le parole date, ognuna "ribaltata" (cioè, stampando i caratteri in ordine inverso: dall'ultimo al primo).

Si supponga che N non sia noto a priori, ma venga fornito da input. Utilizzare una struttura dinamica.

Esercizio 2.

Si consideri il seguente programma:

```
#include <stdlib.h>

int i=3;

int contatore( int k ) {
static int i=-2;
i=i+k;
// A
return i;
}

void f( int *k ) {
    // B
    *k=contatore( i );
    // C
}

main() {
    int i=2, j=3;
    // D
    f( &j );
    // E
    if( i>j ) {
        int j;
        i=1;
        j = contatore( i );
        // F
    }
    // G
}
```

Si indichi, per ciascun punto (A, B, C, ...), quali variabili sono visibili, quali sono presenti in memoria, e per quelle presenti in memoria si indichi che valore hanno.

Esercizio 3.

Si implementino i seguenti punti:

1. Si definisca nell'area dati globale un vettore `W` di 15 interi.
2. Si definisca una funzione `init_vec` che inizializza `W` con 15 valori forniti da input.
3. Si definisca una funzione `out_vec` con i seguenti requisiti:
   - `out_vec` ha tra i propri parametri **due interi**, `x` e `n`;
   - quando viene invocata con `n`$\leq$15, `out_vec` restituisce al chiamante **un vettore** `V`, composto prendendo **solo gli elementi dispari** tra i primi `n` di `W`, ciascuno moltiplicato per `x`.
   - quando viene invocata con `n`$>$15, `out_vec` restituisce al chiamante un valore di errore.
4. Si implementi un `main` che chiami `out_vec` con opportuni parametri e poi stampi su video il contenuto del vettore costruito da `out_vec` nel caso di successo, o un messaggio di errore in caso contrario.

Esercizio 4.

Si implementi una funzione `calculate` con i seguenti requisiti:

1. ha 3 parametri di input: due `int` e un `char`;
2. restituisce uno tra i seguenti risultati:
   - la somma tra i 2 interi, se il `char` vale `'p'`;
   - la differenza, se il `char` vale `'m'`;
   - la differenza tra il numero di somme e il numero di sottrazioni eseguite, se il `char` vale `'?'`.

SOLUZIONI

Esercizio 1.

```
#include <stdio.h>
#include <stdlib.h>

typedef char stringa[21];

stringa *init_V( int* );
void stampa( stringa*, int );
void stampa_reverse( stringa );

int main () {
    stringa *V;
    int dim;

    V=init_V( &dim );
    if( V!=NULL ) {
        stampa( V, dim );
        free( V );
    }

    return 0;
}

/*
 * crea una struttura dinamica
 * inizializza i valori da input
 * restituisce la dimensione tramite n
 * output=puntatore alla struttura
 * output=NULL in caso di errore
 */
stringa *init_V( int *n ) {
    int i;
    stringa *V=NULL;

    printf( "Quante stringhe? " );
    scanf( "%d", n );
    if( n>0 )
        V=( stringa* )malloc(
            ( *n )*sizeof( stringa )
                          );
    if( V!=NULL )
        for( i=0; i<( *n ); i++ )
            scanf( "%s", V[i] );
    return V;
}

/*
 * stampa al contrario le prime dim
 * stringhe del vettore V
 */
void stampa( stringa *V, int dim ) {
    int i;
    for( i=0; i<dim; i++ )
        stampa_reverse( V[i] );
}

/*
 * stampa una stringa al contrario
 * ipotesi: la stringa ha il terminatore
 */
void stampa_reverse( stringa S ) {
    int i=0;
    while( S[i++]!='\0' );
    while( i-->0 )
        putchar( S[i] );
    putchar( '\n' );
}
```

Esercizio 2.

- (`main:D`) Visibili:

  `i`← 2 (`int`) (locale (`main`), *stack*),

  `j`← 3 (`int`) (locale (`main`), *stack*).

  Invisibili (ma presenti):

  `i`← 3 (`int`) (globale, *data segment*),

  `i`← -2 (`int`) (`static` (`cont...`), *data s.*).

- (`f:B`) Visibili:

  `k`← &$\texttt{j}_{\texttt{main}}$ (`int*`) (locale (`f`), *stack*),

  `i`← 3 (`int`) (globale, *data s.*),

  Invisibili (ma presenti):

  `i`← -2 (`int`) (`static` (`cont...`), *data s.*).

  `i`← 2 (`int`) (locale (`main`), *stack*),

  `j`← 3 (`int`) (locale (`main`), *stack*).

- (`contatore:A`) Visibili:

  `i`← 1 (`int`) (`static` (`cont...`), *data s.*).

  `k`← 3 (`int`) (locale (`cont...`), *stack*),

  Invisibili (ma presenti):

  `i`← 3 (`int`) (globale, *data s.*),

  `k`← &$\texttt{j}_{\texttt{main}}$ (`int*`) (locale (`f`), *stack*),

  `i`← 2 (`int`) (locale (`main`), *stack*),

  `j`← 3 (`int`) (locale (`main`), *stack*).

- (`f:C`) Visibili:

  `k`← &$\texttt{j}_{\texttt{main}}$ (`int*`) (locale (`f`), *stack*),

  `i`← 3 (`int`) (globale, *data s.*),

  Invisibili (ma presenti):

  `i`← 1 (`int`) (`static` (`cont...`), *data s.*).

  `i`← 2 (`int`) (locale (`main`), *stack*),

  `j`← 1 (`int`) (locale (`main`), *stack*).

- (`main:E`) Visibili:

  `i`← 2 (`int`) (locale (`main`), *stack*),

  `j`← 1 (`int`) (locale (`main`), *stack*).

  Invisibili (ma presenti):

  `i`← 3 (`int`) (globale, *data s.*),

  `i`← 1 (`int`) (`static` (`cont...`), *data s.*).

- (blocco `main:F`) Visibili:

  `i`← 1 (`int`) (locale (`main`), *stack*),

  `j`← 2 (`int`) (locale (blocco `main`), *stack*).

  Invisibili (ma presenti):

  `j`← 1 (`int`) (locale (`main`), *stack*).

  `i`← 3 (`int`) (globale, *data s.*),

  `i`← 2 (`int`) (`static` (`cont...`), *data s.*).

- (`main:G`) Visibili:

  `i`← 1 (`int`) (locale (`main`), *stack*),

  `j`← 1 (`int`) (locale (`main`), *stack*).

  Invisibili (ma presenti):

  `i`← 3 (`int`) (globale, *data s.*),

  `i`← 2 (`int`) (`static` (`cont...`), *data s.*).

Esercizio 3.

```
#include <stdio.h>
#define DIM 15

// variabili globali
int W[DIM];

// dichiarazioni di prototipi

/*
 * init_vec( V, dim )
 * - inizializza un vettore di interi V di
 * dimensione fisica dim
 * - restitusce la dimensione logica
 * - si interrompe quando l'utente
 * inserisce un valore non intero, o al
 * raggiungimento della dimensione fisica
 */
int init_vec( int*, int );

/*
 * out_vec( A, dim, B, x, n )
 * - crea un vettore B a partire dagli
 * elementi di un vettore A di dimensione
 * logica dim
 * - se n<= 15: B contiene solo gli elementi
 * dispari tra i primi n di A, ciascuno
 * moltiplicato per x, e out_vec restituisce
 * la dimensione logica di B
 * - se n>15: restituisce -1 (errore)
 */
int out_vec( int*, int, int*, int, int );

/*
 * print_vec( V, dim )
 * - stampa il contenuto di un vettore V
 * di dim interi
 */
void print_vec( int*, int );

// definizioni di funzioni
int main() {
    int N, M, x, n, V[DIM];
    N=init_vec( W, DIM );
    scanf( "%*c" ); // svuota buffer input
    printf( "Inserire due x ed n.. " );
    scanf( "%d%d", &x, &n  );
    M=out_vec( W, N, V, x, n );
    if( M!=-1 )
        print_vec( V, M );
    else
        puts( "Errore" );
    return 0;
}

int init_vec( int *V, int dim ) {
    int i=0;
    puts( "Inizializzazione del vettore." );
    printf( "Inserire %d interi\n", dim );
    while( i<dim )
        if( scanf( "%d", &V[i] )>0 )
            i++;
        else
            dim=0;
    return i;
}

int out_vec( int *A, int dim,
             int *B,
             int x, int n ) {
    int i,j=0;
    if( n>DIM )
        return -1;
    if( dim<n )  // ulteriore controllo
        return -1;
    printf( "n: %d, x: %d\n", n, x );
    for( i=0; i<n; i++ )
        if( A[i]%2 )
            B[j++]=x*A[i];
    return j;
}

void print_vec( int *V, int dim ) {
    int i;
    puts( "Contenuto del vettore:" );
```

```
    for( i=0; i<dim; i++ )
        printf( "%d\n", V[i] );
}
```

Esercizio 4.

```
#include <stdio.h>

/*
 * restitusce:
 * --> x+y se c=='p'
 * --> x-y se c=='m'
 * --> la differenza tra il numero di somme
 * e il numero di differenze effettuate, se
 * se c=='?'
 */
int calculate( int, int, char );

int main() { // test
    printf( "calculate( 3, 4, 'p' ): %d\n",
           calculate( 3, 4, 'p' ) );
    printf( "calculate( 4, 2, 'p' ): %d\n",
           calculate( 4, 2, 'p' ) );
    printf( "calculate( 3, 2, 'm' ): %d\n",
           calculate( 3, 2, 'm' ) );
    printf( "calculate( 2, 4, 'p' ): %d\n",
           calculate( 2, 4, 'p' ) );
    printf( "calculate( 3, 4, '?' ): %d\n",
           calculate( 3, 4, '?' ) );
    return 0;
}

int calculate( int x, int y, char c ) {
    static int count=0;
    switch ( c ) {
        case 'p':
            count++;
            return x+y;
            break;
        case 'm':
            count--;
            return x-y;
            break;
        case '?':
            return count;
            break;
        default:
            return 0;
            break;
    }
}
```