

Programmazione Appello d'esame 7 Settembre 2007

Soluzioni (Tema A)

Esercizio 1.1

cegi
1_1_3
5_1_5

Esercizio 1.2

La principale cosa che hanno in comune le stringhe e gli array è l'essere entrambi, in ultima analisi, dei puntatori. Una stringa, infatti, altro non è che un puntatore a char, mentre un array può essere pensato come un puntatore al primo elemento dell'array stesso. Nella fattispecie, però, esiste una differenza importante tra una stringa e un array di caratteri. Questo esempio chiarisce la situazione; si consideri la seguente dichiarazione:

```
char *p = ciao ;
```

Questa è la dichiarazione della variabile p come una stringa contenente la parola *ciao*. Tale dichiarazione è però diversa dalla seguente:

```
char p[3] = {'c','i','a','o'};
```

La prima, infatti, aggiunge automaticamente il carattere \0 che indica la fine della stringa mentre la seconda ne è sprovvista. Quindi, se si esamina la prima dichiarazione il vettore di caratteri da essa costruito, non si troverà {'c','i','a','o'} ma {'c','i','a','o','\0'} .

Esercizio 1.3

Il prototipo di malloc è presente in *stdlib.h* . Esso può essere così descritto:

```
void *malloc (size_m);
```

La malloc viene di solito richiamata passandole come parametro un'espressione del tipo

```
( n * sizeof (tipo_di_dato) )
```

e restituisce un puntatore ad un array di n elementi del tipo *tipo_di_dato* .

Questa funzione è molto utile per la sua flessibilità; essa, infatti, permette di allocare memoria senza conoscere preventivamente la dimensione dell'array.

A differenza della calloc, la malloc non inizializza gli elementi.

Aggiungo un esempio chiarificatore.

```

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

main ()
{

    int n, i;
    int *m;

    /* Richiesta della dimensione dell'array e sua lettura */
    printf ("\nInserisci il numero di elementi dell'array: ");
    scanf ("%d", &n);

    /* Uso della malloc */
    m = malloc ( n * sizeof (int ) );

    /* Scrive l'array da input */
    for ( i = 0; i < n; i++ )
    {
        printf ("\nInserisci l'elemento in posizione %d; ", i);
        scanf ("%d", *(m+i) );
    }

    return 0;
}

```

Esercizio 2.1

```

void CalcolaSerie (double q, int n)
{

    printf ("\nValore di convergenza = %f ", 1/(1-q) );

    int i;

    for (i = 0; i <= n; i++)
        printf ("\n%f", (1-pow(q,i))/(1-q));

    printf ("\n");

}

```

Esercizio 2.5

A)

```

int IsSimmetrica ( float mat[N][N] )
{

```

```

int i,j;

for (i = 0; i < N; i++)
{
    for (j = i+1; j < N; j++)
    {
        if (mat[i][j] != mat [j][i] )
            return 0;
    }
}

return 1;

}

```

B)

```

float Norm ( float v[N] )
{

    float a = 0;
    int i;

    for (i = 0; i < N; i++)
    {
        a += v[i] * v[i];
    }

    return sqrt (a);

}

```