

Laboratorio di Informatica Generale I UD

Settima esercitazione

Danilo Severina

27 Aprile 2006

Funzioni

Definizione della funzione – PRIMA della funzione main()

```
tipo_ritorno nome_funz (tipo_param nome_param_1, ...)  
{  
    ...corpo_funzione...  
}
```

Invocare la funzione da un'altra funzione

```
nome_funz (nome_parametro_1, ...);  
oppure  
var = nome_funz (nome_parametro_1, ...);
```

Esempio:

Scrivere un programma in grado di inizializzare una matrice di interi le cui dimensioni sono indicate dall'utente. Stampare tale matrice.

```
#include <stdlib.h>  
#include <stdio.h>  
  
#define MAX 20  
#define MAX_NUM 90  
#define TRUE 1  
#define FALSE 0
```

```

/* Funzione per l'inizializzazione di una matrice con
numeri casuali compresi tra 0 e MAX_NUM.
Dimensione della matrice MAX x MAX.
Il numero di righe e di colonne effettivamente utilizzate
sono indicati rispettivamente dai valori nrow e ncol.
Return:
TRUE se la matrice e' stata inizializzata corretteamente
FALSE se la dimensione nrow o ncol e' maggiore di MAX
*/

```

```

int inizializza(int m[MAX][MAX], int nrow, int ncol)
{
    int i, j;
    if(nrow>MAX || ncol> MAX)
        return FALSE;
    srand((unsigned)time(NULL));
    for (i=0; i<nrow; i++)
        for (j=0; j<ncol; j++)
            m[i][j]=rand()%(MAX_NUM+1);
    return TRUE;
}

```

```

/* Funzione per la stampa di una matrice.
Dimensione della matrice MAX x MAX.
Il numero di righe e di colonne effettivamente utilizzate
sono indicati rispettivamente dai valori nrow e ncol.
*/

```

```

void stampa(int m[MAX][MAX], int nrow, int ncol)
{
    int i, j;
    for (i=0; i<nrow; i++)
    {
        for (j=0; j<ncol; j++)
            printf("%2d ", m[i][j]);
        printf("\n");
    }
}

```

```

int main()
{
    int A[MAX][MAX], i, j, nr, nc, s1, s2;
    int successo;

    printf("\nInserisci numero righe ");

```

```

scanf("%d", &nr);
printf("\nInserisci numero colonne ");
scanf("%d", &nc);

successo=inizializza(A, nr, nc);

if(successo)
    stampa(A, nr, nc);
else
    printf("Errore: dimensioni non corrette.\nFine.\n");

return(0);
}

```

Esercizi

Esercizio 7.1

Scrivere un programma in grado di scambiare il contenuto di due variabili.
Funzione swap o scambio.

Esercizio 7.2

Scrivere un programma in grado di chiedere in input una stringa alfanumerica e di generare una serie di possibili anagrammi (senza significato). Il numero di anagrammi è richiesto dall'utente.

(Vedi esercizio 6.1)

Struttura soluzione:

funzione main

- richiesta all'utente di
 - stringa da anagrammare (stringa)
 - quanti anagrammi (num)
- per ogni anagramma richiesto (num)
 - invocare la funzione anagramma
- fine

funzione anagramma

- generazione di un numero casuale di permutazioni (p)
- per ogni permutazione prevista
 - scambio di due caratteri
- stampa a video della stringa
- fine

Esercizio 7.3

Scrivere un programma in grado di ricercare all'interno di una matrice $N \times M$ di caratteri una stringa e di indicarne il punto di inizio (coordinante riga-colonna) e la direzione (orizzontale o verticale). La matrice viene inizializzata in modo casuale e la stringa di caratteri si può trovare in orizzontale (da sinistra destra) o in verticale (dall'alto in basso).

La lunghezza l della stringa deve essere minore della minor dimensione della matrice:

$$l \leq \min(N, M)$$

Soluzione con funzioni.

(Vedi esercizio 6.6)

Possibile struttura soluzione:

funzione main

- richiesta all'utente di
 - numero righe matrice
 - numero colonne matrice
 - stringa da ricercare
- invocare funzione di inizializzazione matrice
- per ogni elemento della matrice $m[i][j]$
 - invocare la funzione ricerca in riga con inizio in $m[i][j]$
 - invocare la funzione ricerca in colonna con inizio in $m[i][j]$
- fine

funzione inizializzazione matrice

- inizializzare matrice con caratteri casuali
- fine

funzione ricerca in riga

- se la posizione corrente ed i caratteri successivi (a destra) corrispondono alla stringa da ricercare
 - stampa posizione
- fine

funzione ricerca in colonna

- se la posizione corrente ed i caratteri successivi (in giu') corrispondono alla stringa da ricercare
 - stampa posizione
- fine

Esercizio 7.4

Scrivere un programma in grado di effettuare somma, sottrazione e moltiplicazione di due numeri complessi. Il programma deve essere in grado di richiedere all'utente due numeri complessi e l'operazione da eseguire sui due numeri e successivamente di fornire il risultato dell'operazione.

Si devono utilizzare tre diverse funzioni che implementano le operazioni possibili.

I numeri complessi devono essere memorizzati in vettori di float di due elementi.

La funzione somma deve essere definita nel seguente modo:

```
void somma(float n1[2], float n2[2], float ris[2]){...}
```

dove `n1` contiene il primo numero complesso, `n2` contiene il secondo numero complesso e `ris` contiene il risultato dell'operazione.

Le altre funzioni sottrazione e prodotto devono essere definite in modo analogo alla precedente.

Esercizio 7.5

Scrivere un programma in grado di

- inizializzare un vettore con numeri casuali;
- stampare il vettore (dall'elemento di indice 0 all'elemento di indice MAX);
- invertire il vettore;
- stampare il vettore (dall'elemento di indice 0 all'elemento di indice MAX).

L'inversione del vettore avviene con successivi scambi di due elementi: primo e ultimo, secondo e penultimo, terzo e terzultimo...

Utilizzare funzioni in modo che la funzione main sia del tipo

```
int main()
{
    int v[MAX], n;

    printf("\nNumero elementi del vettore: ");
    scanf("%d", &n);
    inizializza(v, n);
    stampa(v, n);
    inverti(v, n);
    stampa(v, n);

    return 0;
}
```

Esercizio 7.6

Scrivere un programma in grado di inizializzare in modo casuale un vettore `v` di MAX=10 interi e successivamente di ordinare lo stesso vettore in ordine crescente (in modo che sia soddisfatta la condizione

$$v[i-1] \leq v[i] \leq v[i+1]$$

con i compreso tra 1 e $MAX-2$).

La funzione main deve risultare:

```
int main()
{
    int v[MAX], n;

    printf("\nNumero elementi del vettore: ");
    scanf("%d", &n);
    inizializza(v, n);
    stampa(v, n);
    ordinamento(v, n);
    stampa(v, n);

    return 0;
}
```

Versione 1. Selection sort.

Per ogni elemento i ($i=0 \dots MAX-2$) (!!) del vettore v si cerca nella restante parte del vettore (il sotto-vettore da $v[i+1]$ a $v[MAX-1]$) il minimo valore minore di $v[i]$: il valore così trovato lo si scambia con l'elemento $v[i]$.

Versione 2. Bubble sort

*Per ogni elemento i ($i=0 \dots MAX-2$) (!!) del vettore v si analizza **tutta** la restante parte del vettore (il sotto-vettore da $v[i+1]$ a $v[MAX-1]$): se un elemento $v[j]$ (con $j=i+1 \dots MAX-1$) è minore di $v[i]$ si effettua lo scambio poi si continuano i confronti fino a che l'indice j non ha raggiunto il suo valore massimo.*