LABORATORIO DI PROGRAMMAZIONE 1

Docente: dr. Vincenzo Bonnici

Lezione 13 – 13/03/2015

FUNZIONI: VARIABILI LOCALI E GLOBALI

- Le variabili definite all'interno di una funzione sono dette *locali*.
  - Il valore di una variabile locale può essere utilizzato solo all'interno della funzione in cui la variabile è definita.

- Le variabili definite all'esterno di qualsiasi funzione dette *globali*.
  - Il valore di una variabile globale può essere utilizzato da qualsiasi funzione del programma: qualsiasi funzione può modificarne il valore.
  - Le variabili globali sono definite per prime nel programma.

FUNZIONI: VARIABILI LOCALI E GLOBALI

```
#include <stdio.h>
int multiplier;                    // variabile globale

int mult( int a ){
  return a * multiplier;           // uso variabile globale
}

void changeMultiplier( int m ){
  multiplier = multiplier + m;     // modifica variabile globale
}

int main( void ){
  int a = 5;
  changeMultiplier( 10 );
  printf( "%i * %i = %i\n", a, multiplier, mult( a ) ); // 5*10
  changeMultiplier( 8 );
  printf( "%i * %i = %i\n", a, multiplier, mult( a ) ); // 5*18
  return 0;
}
```

FUNZIONI: VARIABILI LOCALI E GLOBALI

- Le variabili globali possono essere utilizzate nei programmi in cui più funzioni devono accedere al valore di una stessa variabile.
  - Permette di evitare il passaggio esplicito di un parametro.
  - Limita la leggibilità del programma.
  - Riduce la genericità della funzione.

- Valore di default:
  - Il valore di default di una *variabile globale* è zero
    - int a; → a = 0
    - int array[10] → tutti gli elementi sono inizializzati a zero.
  - Le *variabili locali* non hanno un valore iniziale di default e quindi devono essere esplicitamente inizializzate nel programma.

FUNZIONI: VARIABILI AUTOMATICHE E STATICHE

- Le variabili locali dichiarate all'interno di una funzione sono dette *automatiche*:
  - Una variabile automatica è creata ogni volta che la funzione viene creata.
  - Quando la funzione termina, queste variabili "scompaiono".

- Le variabili locali possono essere dichiarate statiche, usando la parola chiave `static` davanti alla dichiarazione della variabile.
  - Una variabile locale statica è creata una sola volta quando viene avviato il programma principale.
  - Quando la funzione termina, la variabile mantiene il suo valore per una prossima invocazione della funzione.
  - Le variabili locali statiche hanno una valore di default pari a zero.

FUNZIONI: VARIABILI AUTOMATICHE E STATICHE

```
#include <stdio.h>

void auto_static( void ){
  int a = 1;
  static int b = 1;          // variabile statica
  printf( "Variabile automatica: %i, variabile statica: %i\n",
          a, b );
  a = a+1;
  b = b+1;
}

int main( void ){
  int i;
  for( i = 0 ; i < 10; i++ ){
    auto_static(); // a sempre 1, b valore da 1 a 10
  }
  return 0;
}
```

FUNZIONI: VARIABILI AUTOMATICHE E STATICHE

- Le variabili statiche sono usate per mantenere il valore di una variabile tra diverse invocazioni della stessa funzione.

- Le variabili statiche sono particolarmente utili quando una funzione usa una variabile il cui valore viene impostato una sola volta e non cambia più.
  - Si evita l'inefficienza di dover reinizializzare la variabile ogni volta. Questo è particolarmente utile con gli array.

ESERCIZI

- Generare numeri random (casuali)

```
srand (time(NULL)); //inizializziamo il generatore random
int length = 10;
int array[length];
int i;
for(i = 0; i < length; i++){
   array[i] = rand() % 10;
   //genera un numero casuale da 0 a 9
}
```

ESERCIZIO 1

- Si scriva un programma che:
  - genera un array monodimensionale di 100 elementi
  - riempie l'array con numeri casuali da 0 a 100
  - tramite la funzione `verifica(...)`, verifica se esiste un elemento tale che esso è maggiore o uguale della somma degli elementi a seguire e ne stampi a video il valore e la posizione

ESERCIZIO 2: PRODOTTO TRA MATRICI QUADRATE

- Data una matrice A di dimensione n×n, ed una matrice B di dimensione n×n
  - il risultato del prodotto matriciale A×B è una nuova matrice di dimensioni n×n tale che
    - $C[i][j] = \sum_{r=0}^{n-1} (A[i][r] * B[r][j])$

- Si scriva un programma che:
  - costruisce due matrici di dimensione 10×10
  - le riempie con numeri casuali da 0 a 50
  - calcola il prodotto matriciale
  - stampa il risultato

ESERCIZIO 3

- Si scriva un programma che:
  - costruisca un array monodimensionale di dimensione 100
  - lo riempie con numeri casuali da 0 a 10
  - tramite la funzione `verifica_crescente(...)`, verifica se esistono all'interno dell'array tre elementi successivi ordinati in modo crescente
  - tramite la funzione `verifica_decrescente(...)`, verifica se esistono all'interno dell'array tre elementi successivi ordinati in modo decrescente
  - tramite la funzione `massimo_crescente(....)`, restituisca il numero massimo di elementi successivi all'interno dell'array che risultano essere ordinati in modo crescente, e li stampi a video

ESERCIZIO 4

- Si scriva un programma che:
  - dati due array monodimensionali, A e B, di dimensione a piacere
  - crea un terzo array C di lunghezza pari alla somma delle lunghezze di A e B, tale che C contiene gli elementi di A e B disposti in ordine crescente
  - NB: non si possono utilizzare algoritmi di ordinamento

ESERCIZIO 5

Il Crivello di Erastotene è una tecnica per calcolare i numeri primi.

- Si disegna una tabella 10×10 con i numeri da 1 a 100
- I numero possono essere liberi o cancellati
- Inizialmente tutti i numeri sono liberi, eccetto l'1 che è cancellato (da che non è un numero primo, per definizione)
- Si sceglie il primo numero libero (il 2) e si cancellano nella tabella tutti i suoi multipli
- Si sceglie il successivo numero libero, e si cancellano a sua volta tutti i suoi multipli
- Si procedere nello stesso modo fino alla fine della tabella
- I numeri che restano non cancellati sono tutti e soli i numeri primi tra 0 e 100