Informatica ed Elementi di Informatica Medica

A.A. 2018-19

Laboratorio n° 4

Ing. Gian Enrico Conti

Info Logistiche

- Contatti:
  - gianenrico.conti@mail.polimi.it

- Sito web del laboratorio:
  - http://home.deib.polimi.it/barenghi/doku.php?id=teaching:ieim#materiale_laboratorio

- **Nota per le mail:**

  Oggetto: *[INFO-BIO] il vostro oggetto*

Info Logistiche: Calendario laboratori

DATA	Orario	Squadra	Aula	Programma
15/3 Ven	14:15 alle 17:15	A	L.26.14	LAB1
18/3 Lun	15:15 alle 18:15	B	"	LAB1
22/3 Ven	14:15 alle 17:15	A	"	LAB2
25/3 Lun	15:15 alle 18:15	B	"	LAB2
29/3 Ven	14:15 alle 17:15	A	"	LAB3
1/4 Lun	15:15 alle 18:15	B	"	LAB3
5/4 Ven	14:15 alle 17:15	A	"	LAB4
8/4 Lun	15:15 alle 18:15	B	"	LAB4
12/4 Ven	14:15 alle 17:15	A	"	LAB5
29/4 Lun	15:15 alle 18:15	B	"	LAB5

Rubrica contatti recap

- Struct
- array di struct
- f. Che ritornano un “contatto”
- f. Di stampa stringhe (for..)
- f. Di stampa interi
- f. Di stampa di struct

Esercizio 1: Funzioni

Scrivere un programma che implementi la
f. Numero primo

La f. Deve restituire o se NON primo, 1 se primo

Puntatori

I **puntatori** sono delle variabili che contengono l'indirizzo di memoria di un'altra variabile

```
int* puntatore;
```

"*" è chiamato **operatore di deferenziazione** e restituisce il contenuto dell'oggetto puntato dal puntatore; mentre l'operatore "&" restituisce l'indirizzo della variabile.

```
int var=0, var2, ind;
int* puntatore;
puntatore = &var; //puntatore punta a var
var2 = *puntatore; //var2 == var
ind = puntatore; //ind contiene l'indirizzo di var
*puntatore = 5 //var == 5
```

Operazioni sui puntatori

- È possibile effettuare operazioni sui puntatori.
  - Spostamento in avanti: *p* + *i*, sposta il puntatore di *i* posizioni avanti
  - Spostamento indietro: *p* – *i*, sposta il puntatore di *i* posizioni indietro
- **Un puntatore esiste sempre in funzione del tipo di oggetto puntato.**
  - Un puntatore ad *int* ha un blocco di memoria di 4 byte, a *char* di 1 byte, ecc...
  - Se sommo al puntatore un intero o utilizzo l'operatore **++,** il puntatore si sposterà in avanti di tanti byte quanti ne prevede il tipo di variabile puntata. Ad esempio: *p* + *i* equivale a *posizione* + *(i*dimensione tipo puntato)*
- Se ad una funzione passo dei puntatori eventuali operazioni su questi parametri saranno effettuate sulle variabili originali!
- È possibile agire sugli array come se si stesse agendo su un puntatore poiché entrambi sono blocchi contigui di memoria. **Attenzione:** devo comunque assegnare la prima cella dell'array ad un puntatore.
- Posso utilizzare i puntatori per puntare a strutture, in questo caso per accedere ai campi della *struct* utilizzo l'operatore **"->"**

Puntatori e tipi

- Puntatori ed array

```
char stringa[20];
char* pointer;
pointer= &stringa[0]; //pointer punta a stringa[0] pointer++; //Ora pointer
punta a stringa[1]
```

- Puntatori e struct

```
struct punto {
    int x;
    int y;
} puntoA;

struct punto* pointer;
pointer = &puntoA;
pointer->x = 6; //Assegno il campo x
pointer->y = 7; //Assegno il campo y
```

Esercizio 1: Somma tra matrici

Scrivere un programma che date due matrici chiami una funzione ne calcoli la matrice somma.

```
void somma_m(int m1[][], int m2[][], int m_ris[][]);
```

Le matrici devono avere stesse dimensioni tra di loro

Esercizio 2: Somma con puntatori

Scrivere una funzione che calcoli la somma di due interi e salvi il risultato in una variabile passata come puntatore.

```
void sommap(int num1, int num2, int* ris);
```

Esercizio 3: Funzioni I/O e puntatori

Si scriva una funzione che legga da tastiera una stringa fino alla pressione del tasto invio.
La stringa da caricare e' passata per indirizzo:

firma:

```
int leggiStringa(char *s);
```

Hint:
- si usi getchar()
- la f. Ritorna il n. Di caratteri letti

Esercizio 4: colori

In Linux / Unix possiamo scrivere a colori:

```
// colors:
// https://bluesock.org/~willkg/dev/ansi.html
```

Es:

```
void myprint(char s[]){
     int i;
     for (i=0; s[i]!=0 ; i++) {
          putchar(s[i]);
     }
}
int main () {
     myprint("\033[1;31m");
     myprint("Hello world\n");
     myprint("\033[0m;");
     return 0;
}
```

```
Last login: Fri Apr  5 08:32:59 on ttys002
macbookpro-32mb:~ ingconti$ /Users/ingconti/
ild/Products/Debug/writeInColor ; exit;
Hello world
```

Esercizio 5: Mastermind

Scrivere un programma per giocare a **Mastermind**.

http://www.webgamesonline.com/mastermind/ http://www.webgamesonline.com/mastermind/rules.php

Esercizio 5: Mastermind (cont.)

Regole di base:

- 2 giocatori (CPU e utente)
- CPU sceglie una sequenza di colori non nota all'utente
- L'utente deve indovinare l'esatta sequenza (posizione e colore)
- Ad ogni round l'utente inserisce una possibile sequenza
- CPU risponde in questo modo:
  - Per ogni posizione e colore indovinato nella sequenza, mostra un colore nero a destra (=black peg)
  - Per ogni colore indovinato (ma posizione sbagliata) ella sequenza, mostra una colore bianco (=white peg)

Esercizio 5: Mastermind (cont. 2)

Altre regole:

- L’utente definisce la lunghezza della sequenza da indovinare
- Nella sequenza i colori si possono ripetere
- L’utente definisce il numero massimo di round in cui indovinare la sequenza
- L’utente vince se indovina l’esatta sequenza entro il numero massimo di round, altrimenti vince CPU.

Esercizio 5: Mastermind (cont. 3)

Requisiti:

- L’utente deve definire le impostazioni di gioco (lunghezza sequenza, numero massimo di round)
- CPU deve generare una sequenza casuale di colori
- L’utente deve poter inserire una possibile sequenza ad ogni round
- CPU deve mostrare la qualità del risultato del round (colori bianchi e/o neri)

Hints:

- Utilizzare i codici colori visti es. 4 (oppure un solo char i.e. ‘w’ x white)
- Definire un carattere per ogni colore per l’inserimento (e.g., ‘g’ = "giallo")
- 6 colori: rosso, giallo, verde, arancione, nero e bianco (provarli prima..)

- ***SUDDIVIDERE la logica in funzioni FIN DALL’ INIZIO della scrittura del programma.***