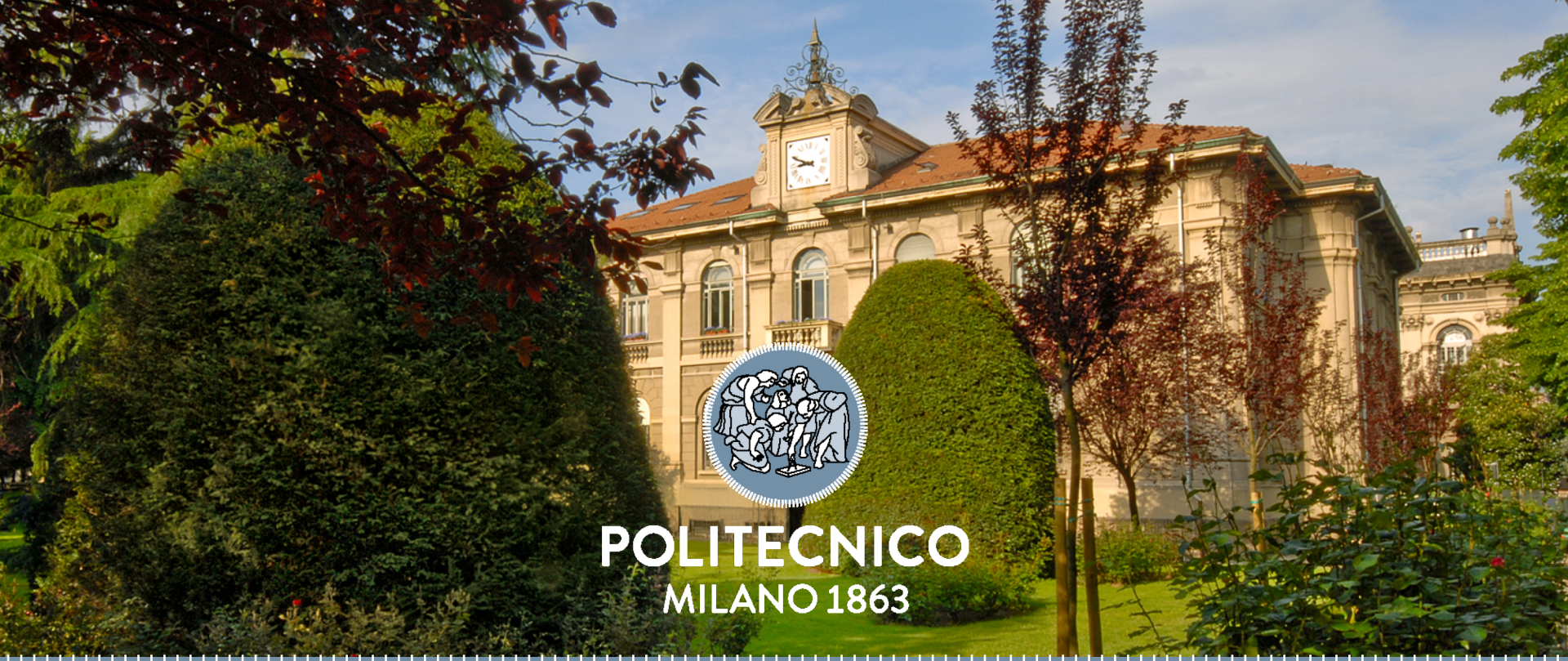

Informatica ed Elementi di Informatica Medica

A.A. 2017-18

Laboratorio n°3

Dott. Michele Zanella

Ing. Gian Enrico Conti

HEAP LAB

Info Logistiche (cont'd)

- Calendario laboratori

Data	Orario	Squadra	Aula	Resp.	Programma
16/03/18	14:15-17:15	A	L26.14	Conti	Lab 1
19/03/18	15:15-18:15	B	L26.14	Zanella	
23/03/18	14:15-17:15	A	L26.14	Conti	Lab 2
26/03/18	15:15-18:15	B	L26.14	Zanella	
06/04/18	14:15-17:15	A	L26.14	Conti	Lab 3
09/04/18	15:15-18:15	B	L26.14	Zanella	
13/04/18	14:15-17:15	A	L26.14	Conti	Lab 4
16/04/18	15:15-18:15	B	L26.14	Zanella	
20/04/18	14:15-17:15	A	L26.14	Conti	Lab 5
23/04/18	15:15-18:15	B	L26.14	Zanella	

Esercizio 3.1: Matrice - Riempimento sequenziale

Scrivere un programma che riempia una matrice in modo sequenziale, ricevendo gli interi da I/O, e che stampi la matrice formattata.

Hint:

Riutilizzare le putint/getint precedenti.

Esercizio 3.2: Matrice - Riempimento concentrico

Scrivere un programma che riempia una matrice in modo concentrico, ricevendo gli interi da I/O, e che stampi la matrice formattata.

Hint:

Riutilizzare le putint/getint precedenti.

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 3.3: Somma tra matrici

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

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

Le matrici devono avere stesse dimensioni tra di loro

Esercizio 3.4: 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 3.5: Somma vettori con puntatori

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

```
void sommavp(int* v1, int* v2, int* ris);
```

Hint:

Considerate la dimensione dei vettori come prefissata

Esercizio 3.6: Rubrica del telefono

Si scriva un programma che implementi una semplice rubrica del telefono.

Requisiti:

- Possibilità di inserire/cancellare contatti
- Possibilità di stampare elenco contatti con informazioni
- Possibilità di ricercare un contatto dato il cognome

Extra:

Possibilità di effettuare statistiche riguardanti l'età (media, mediana) e nomi dei contatti (numero di occorrenze)

Hints:

- Riutilizzare eventuali funzioni già implementate
- Settare un numero massimo `CONT_MAX` di contatti

Esercizio 3.6: Rubrica del telefono (cont'd)

Contatto:

Definiamo un tipo di dato personalizzato *contatto*. Il tipo deve includere le informazioni relative a:

- Cognome
- Nome
- Età
- Numero di telefono

Ulteriori esercizi

Provare a riscrivere alcuni esercizi con i vettori e le stringhe dei laboratori passati utilizzando le operazioni con i puntatori.