INFORMATICA A

A.A. 2018-19

Laboratorio n°2

Dott. Michele Zanella

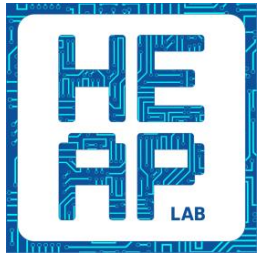
Ing. Gian Enrico Conti

HEAP LAB

Info Logistiche

- Contatti:
  - michele.zanella@polimi.it (Squadra B)
    - HEAP Lab – Campus Leonardo, via Golgi 39, Edificio 21, Piano 1, Ufficio 4, +39 02 2399 **9613**
      (mandatemi una mail per accordarci su giorno e ora)
  - gianenrico.conti@mail.polimi.it(Squadra A e C)
- Sito web del laboratorio:
  - http://zanella.faculty.polimi.it/teaching/informatica-a
- **Nota per le mail:**

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



Cicli

- **For**

```
for(init-expr; test-expr; increment-expr) {
    statement
}
```

  - *Init-expr:* dichiara ed inizializza uno o più contatori
  - *est-expr:* se viene soddisfatta (*true*) allora viene eseguita la *statement*, altrimenti il ciclo termina
  - *Increment-expr:* viene eseguita alla fine di un'iterazione e incrementa uno o più contatori.

```
for(int i=0; i<2; i++)
        printf("%d\t",i);
```

  - **Risultato: *0* *1***

Cicli (cont'd)

- **While**

```
while(condition) {
    statement
};
```

La *statement* viene eseguita finchè la condizione è verificata

- **Do….while**

```
do {
    statement
}while(condition);
```

La condizione viene verificata alla fine, la *statement* è eseguita almeno una volta.

Condizioni

- **If**

```
if(boolean_expression){
    statement
}
```

Lo *statement* viene eseguito se l'espressione è verificata

- **If…else**

```
if(boolean_expression) {
    statement1
} else {
    statement2
}
```

Lo *statement1* viene eseguito se l'espressione è verificata **altrimenti** viene eseguito lo *statement2*

Condizioni

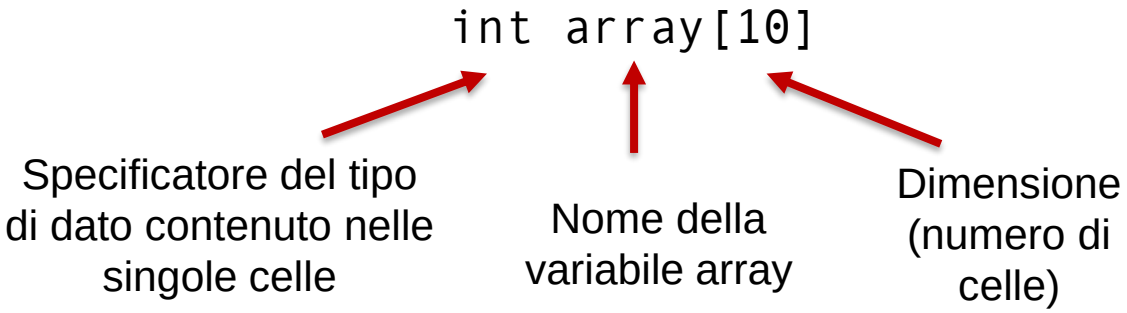
- **Switch**

```
switch(variable) {
    case value:
        statement1;
        …
    default:
        default_statement;
}
```

La variabile viene verificata nei diversi casi, se per uno è vero ne viene eseguito il relativo *statement*. Altrimenti viene eseguito il caso di *default*.

Array

- Sono dei contenitori formati da una serie di celle in numero fissato contenenti valori di un certo tipo (int, long, char, …).



- Accesso agli elementi degli array, se la sua dimensione è 10:
  - Primo elemento: `array[0];`
  - *i*-esimo elemento: `array[i];`
  - Ultimo elemento: `array[9];` ← **ATTENZIONE!**

Stringhe

- Le stringhe sono una sequenza di caratteri. In C esse sono definite come array di *char.*

```
char stringa[50];
```

- Per operare sulle stringhe esiste una libreria *string.h*
- Per stampare una stringa si utilizza lo specificatore di tipo *%s*.
- Per accedere ad un carattere della stringa basta accedere alla posizione nell'array -> `stringa[0]` indica il primo carattere della stringa.
- La stringa viene terminata con il *carattere terminatore \0*

Stringhe: string.h

- *strlen(stringa)*: ritorna la lunghezza della stringa (escluso il carattere terminatore)
- *strcmp(stringa1, stringa2)*: compara le due stringhe, r
  - *< 0 : se stringa1* è alfabeticamente precedente alla *stringa2*
  - *= 0 :* se *stringa1* e *stringa2* sono uguali
  - *> 0 :* se *stringa2* è alfabeticamente precedente alla *stringa1*
- *strcat(stringa_dest, stringa_src):* appende la *stringa_src* alla fine di *stringa_dest*
- *strcpy(stringa_dest, stringa_src)*: copia la *stringa_src* nella *stringa_dest*

Esercizio 2.1: Ricerca sequenziale

Scrivere un programma che, dato un vettore di *n* numeri, ricerchi un valore al suo interno, indicando l'indice del vettore che contiene il numero da ricercare

Esercizio 2.2: Ordinamento – Bubble sort

Scrivere un programma che, dato un vettore non ordinato di *n* numeri, implementi l'algoritmo Bubble sort per ordinare i valori in ordine crescente seguendo.

Bubble sort: Ogni coppia di elementi adiacenti viene comparata e invertita di posizione se sono nell'ordine sbagliato. L'algoritmo continua nuovamente a ri-eseguire questi passaggi per tutta la lista finché non vengono più eseguiti scambi, situazione che indica che la lista è ordinata.

Esercizio 2.3: Sottosequenza

Scrivere un programma che legge una sequenza (di lunghezza arbitraria) di numeri interi positivi, terminata da 0, e indica, alla fine della sequenza, qual è la lunghezza della massima sottosequenza di numeri consecutivi in ordine crescente.

Esempio:

13 3 8 4 5 1 17 0

Lung. Max = 2

21 19 18 14 9 6 4 3

Lung. Max = 1

2 1 3 6 8 0 1 12

Lung. Max = 4

Hint: creare una variable "di stato" che ad ogni iterazione tenga conto della lunghezza massima raggiunta finora ed eventualmente si aggiorni se si incontra una sequenza più lunga.

Esercizio 2.4: Riflessione stringa

Scrivere un programma che, data una stringa, calcoli la sua riflessa.

Esempio:
"roma" -> "amor"

Esercizio 2.5: Alfabeto farfallino

Scrivere un programma che legge da *stdin* una sequenza di caratteri e stampa su *stdout* una sequenza derivata dalla precedente secondo le regole dell'alfabeto farfallino (i.e., ogni vocale viene raddopiiata inserendovi in mezzo una 'f').

Esempio:

quanto mi piace questo corso!

*qu**fu**a**fa**nto**fo** mi**fi** pia**fi**a**fa**ce**fe** qu**fu**e**fe**sto**fo** co**fo**rso**fo**!*

Hint:

- Considerare solo lettere minuscole
- Scandire un carattere alla volta

Varianti:

- La frase sia trascritta interamente in lettere maiuscole -> vedere funzione *toupper()*.

Esercizio 2.6: Confronto tra stringhe

Scrivere un programma che, date due stringhe verifichi se sono uguali o in caso negativo quale viene prima in ordine alfabetico.

Il programma deve stampare:

- 0: le stringhe sono uguali
- -1: la `stringa1` è minore della `stringa2`
- 1: la `stringa2` è maggiore della `stringa2`

Esercizio a casa 2.1: Crivello di Eratostene

Scrivere un programma che implementi il crivello di Eratostene, antico algoritmo per il calcolo dei primi *n* numeri primi.

Esercizio a casa 2.2: Stringa palindroma

Scrivere un programma che, data una stringa (senza spazi), verifica se è palindroma.

Esempio:

"anna" -> è palindroma

"pippo" -> non è palindroma

Esercizio a casa 2.3: Parentesi

Scrivere un programma che richiede all'utente una sequenza di caratteri (stringa) che non contiene spazi, li registra in una struttura dati appropriata e verifica se la sequenza è ben parentesizzata".

- Una sequenza si dice ben "parentesizzata" se le parentesi, per semplicità consideriamo solo quelle tonde, sono aperte e chiuse correttamente.
- Ad esempio sono ben parentesizzate" le sequenze:

  Bla

  (bla)

  (bla(bla))

  (bla()(bla)())

- **Hint:** si tenga conto man mano di quante sono le parentesi già aperte, e si decida di conseguenza. Si arresti la scansione non appena si scopre che la sequenza è illegale, anche prima di arrivare alla fine.