
UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

# 2. Tipi Opachi in C

2 esercizi risolti + 1 extra

Tutorato di
Programmazione Avanzata

RELATORE
Imberti Federico

SEDE
Dalmine, BG

# Key-points della teoria: Decoupling

1. I **tipi opachi** vengono usato in C per implementare il **decoupling** tra la definizione e l'uso di tipi di dato personalizzati, permettendo di:
    a. Favorire un design modulare;
    b. Aumentare la manutenibilità del codice, permettendo anche di cambiare l'implementazione in maniera trasparente per gli utenti;
    c. Rendere riutilizzabile parti di codice;
    d. …
2. Precursore dell'incapsulamento e delle classi private nei linguaggi **object-oriented** (C è procedurale!)

# Key-points della teoria: Decoupling 🚀

# Key-points della teoria: Implementare decoupling

Procedura in 3 step:

1. Definire tipi di dato personalizzati (typedef) e l'**interfaccia** dei metodi senza implementarli (file **.h**);

2. Implementare i tipi personalizzati e il corpo delle funzioni definite nel file di header (file **.c** scritto da chi sviluppa la libreria);

3. Usare i metodi definiti al punto precedente (file **.c** creato dall'utente della libreria).

# Key-points della teoria: persona.h

```
#ifndef                                 persona_h
#define persona_h

    //Definizione di “personaType”

    typedef struct personaType *persona;


    //Metodi di persona

    persona costruttore(char* nome);

    void printNome(persona* p);

#endif
```

# Key-points della teoria: persona.c

```c
#include “persona.h”
struct personaType{
     char *nome;
}
persona costruttore(char* nome){
     persona p = malloc(sizeof(persona));
     . . .
}
void printName(persona* p){ . . . }
```

# Key-points della teoria: main.c

```
. . .
#include “persona.c”
int main(int argc, char const *argv[]) {
    char* nome[] = “Luca”;
    persona p = costruttore(nome);
    printNome(p);
}
```

UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

# Domandine?

# Esercizio 1/2

Definisci il tipo opaco CircularString che rappresenta una stringa circolare di una data lunghezza fissa e con contenuto fisso (ma che può girare).
**costruttore**: crea una CircularString e la inizializza alla stringa passata come argomento (facendone una copia).
**gira**: prende una CircularString e fa scorrere tutti i caratteri di una posizione in avanti e il carattere in ultima posizione diventa il primo. Esempio: gira("pippo") = "opipp". La CircularString tiene anche memoria del numero di giri che sono stati applicati
**toString** che restituisce una stringa (array di caratteri) che rappresenta la CircularString
**n_gira** che restituisce il numero di volte che il gira è stato applicato
**cancella** che distrugge la CircularString e libera la memoria
Fai un esempio in cui:

- costruisci la CircularString "ciao"
- fai circolare la string un paio di volte e ogni volta stampi la stringa e il numero di gira
- cancelli la string

# Esercizio 2/2

Definisci il tipo opaco StringBuffer che rappresenta un array di caratteri a cui possono accodare altri array di caratteri (stringhe). La lunghezza dell'array deve aumentare se necessario e deve aumentare la memoria a disposizione).

**costruttore**: crea una StringBuffer e lo inizializza alla stringa passata come argomento (facendone una copia).

**accoda**: prende un StringBuffer X e un altro StringBuffer Y e accoda Y a X cambiandone lunghezza.

**toString** che restituisce una stringa che rappresenta lo StringBuffer. Deve mettere all'inizio e alla fine un "

**cancella** che distrugge StringBuffer e libera la memoria

Fai un esempio in cui:

- costruisci la StringBuffer con l'array "ciao" e lo stampi
- accodi l'array " mario" e lo stampi
- cancelli l'array

# Esercizio Extra

Definisci il tipo opaco List che rappresenta una lista di interi. Definisci le seguenti operazioni:

**costruttore**: crea una lista ed inserisce l'insieme di interi contenuti in un array passato come argomento (facendone una copia).
**concatena**: concatena la lista con un'altra lista e restituisce una nuova lista contenente la loro concatenazione.
**print:** stampa la lista.
**cancella**: distrugge la lista e libera la memoria

Implementa la lista mediante una lista puntata.

Fai un esempio in cui:

- Costruisci la lista con gli elementi {1,2,3} e lo stampi.
- Calcola la concatenazione delle liste contenenti gli elementi {1,4,5,7} e {8,4,3,1}.
- Cancelli tutte le liste.