PROGRAMMAZIONE II(canale A-D)a.a.2004/2005

Prof.ssa **EMANUELA FACHINI**,
STUDIO: **D.to Informatica, Via Salaria,113**
TEL. **0649918314**
E-MAIL: fachini@di.uniroma1.it
ORARIO DI RICEVIMENTO: **giovedì ore 15 - 17 o anche in coda alle lezioni, quando possibile**.

pagina del corso:
http://twiki.dsi.uniroma1.it/twiki/view/Programmazione2ad/WebHome

Esercitazioni: dott.ssa Irene Finocchi

TESTO CONSIGLIATO:
Al Kelley, I. Pohl C, didattica e programmazione, Addison-Wesley, traduzione italiana a cura del Prof. G. Pighizzini, Univ. Milano

Nel passato sono stati prodotti moltissimi programmi che

- **terminano imprevedibilmente (crash),**
- **non producono risposte corrette (pieni di bugs),**
- **sono costosi da correggere o modificare, e**
- **contengono errori nascosti (timebombs) che si manifesteranno prima o poi**

PROGRAMMAZIONE II

Requisiti dei programmi

• **correttezza**	**proprietà oggettiva**
• **efficienza**	**proprietà oggettiva**
• **facilità di lettura, e quindi comprensione**	**non quantificabile**
• **facilità di manutenzione**	**“**
• **robustezza**	**“**
• **portabilità**	**“**
• **riusabilità**	**“**

L'obiettivo più difficile (!) è scrivere programmi corretti

Cosa fare :

Progettare accuratamente il programma scegliendo le strutture dati più convenienti, suddividendolo in moduli coerenti per obiettivo (per esempio tutte le funzioni di I/O,...)
Per ogni compito che deve essere eseguito scrivere la relativa funzione in pseudocodice e verificarne la correttezza

Cosa non fare:

Scrivere il programma senza preoccuparsi dello stile o della strutturazione e affidare ogni verifica della correttezza al testing

I programmatori professionisti seguono un insieme di regole (una metodologia) che li assiste nella costruzione di buoni programmi.
Una delle più recenti strategia di progetto di programmi è l'

Object orientation

In questo corso adotteremo questa metodologia orientata agli oggetti

L'idea di classe è il concetto chiave della progettazione OO: consiste in una descrizione degli oggetti che si vuole manipolare

Esempio: La classe rettangolo
gli oggetti della classe sono i singoli rettangoli.

La classe è caratterizzata da
1. attributi degli oggetti della classe

Esempio
Prendiamo come attributi del rettangolo altezza e larghezza

e da
2. metodi (cioè funzioni) che consentano di

- **creare e distruggere oggetti della classe**
- **ottenere valori di attributi di oggetti**
- **modificare valori di attributi di oggetti**
- **trasformare oggetti di una classe in oggetti di una classe differente**

Esempio
Prendiamo come metodi del rettangolo:

Nome	Descrizione
CostRettangolo	crea un rettangolo
DistrRettangolo	distrugge un rettangolo
Larg	ottiene la larghezza
Alt	ottiene l'altezza
ModAlt	modifica la larghezza
ModLarg	modifica l'altezza

Proprietà

Perimetro	calcola il perimetro
Area	calcola l'area

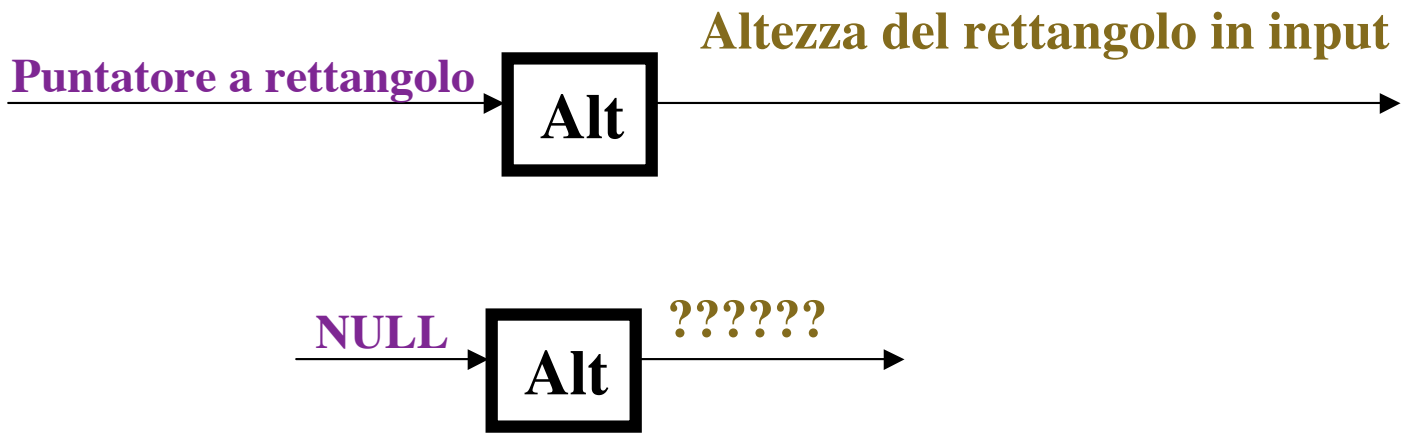
La specificazione formale in C della classe è quindi:

/*il nome della classe è quello del tipo di dato
utilizzato per rappresentare gli oggetti in memoria*/

```
typedef struct rettangolo *RettangoloP;

        RettangoloP CostRettangolo( double alt, double larg );
        void DistrRettangolo( RettangoloP r );
        double Alt( RettangoloP r );
        void ModAlt( RettangoloP r, double alt );
        double Larg( RettangoloP r );
        void ModLarg( RettangoloP r, double larg);
        double Perimetro( RettangoloP r );
        double Area(RettanoloP r);
```

double Alt(RettangoloP r);



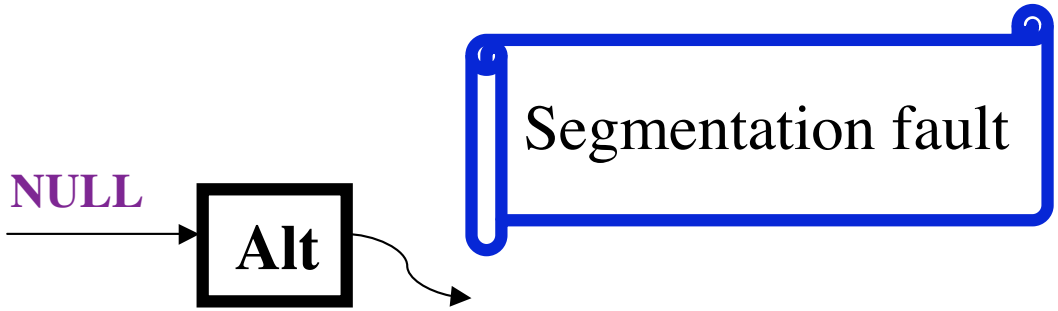
Scelta 1.

```
double alt(RettangoloP r)
/*restituisce l'altezza di r, per r != NULL */
```

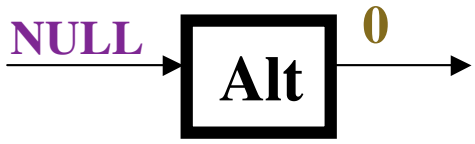
Scelta 2.

```
double alt(RettangoloP r)
/* restituisce l'altezza di r se r!=NULL, altrimenti 0.*/
```

Scelta 1.

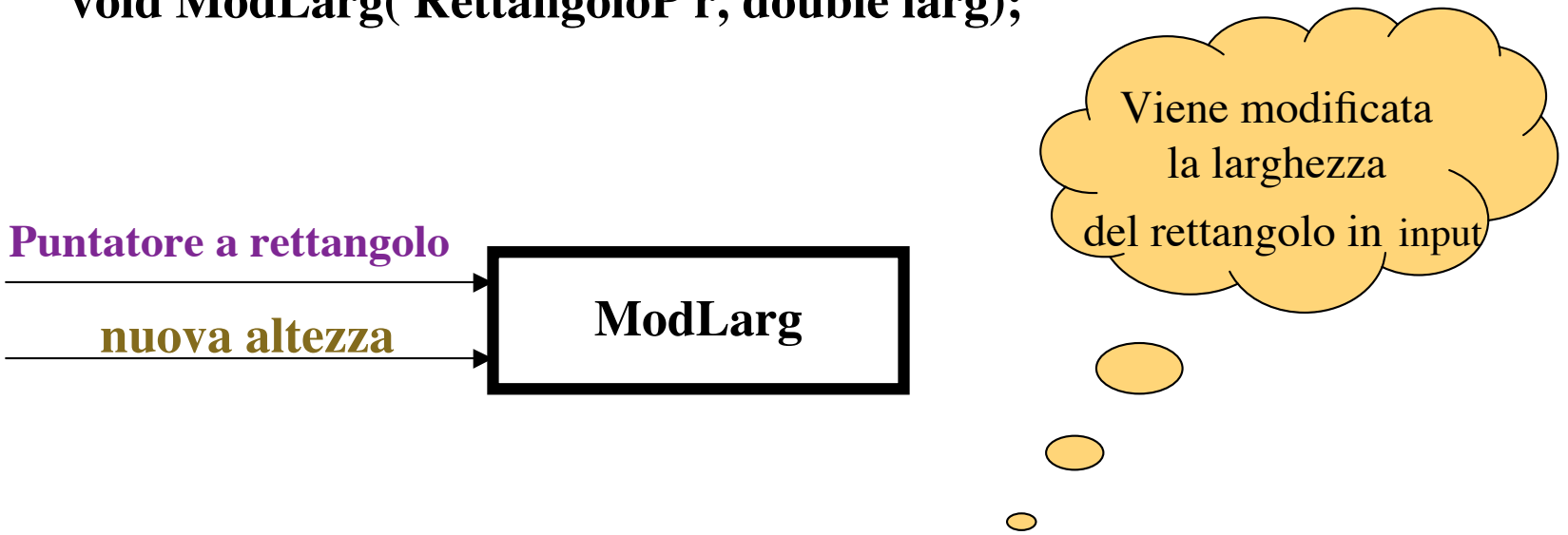


Scelta 2.



Ma codice meno efficiente!!

void ModLarg(RettangoloP r, double larg);



Oltre a dover gestire il caso del puntatore nullo, si deve decidere come trattare il caso in cui il valore passato per altezza è negativo o nullo