TIPI DI DATO ASTRATTO

Un *tipo di dato astratto (ADT)* definisce una categoria concettuale con le sue proprietà:

- una ***definizione di tipo***
  - **implica un dominio, D**
- un ***insieme di operazioni ammissibili* su oggetti di quel tipo**
  - **funzioni: *calcolano valori sul dominio D***
  - **predicati: *calcolano proprietà vere o false su D***

TIPI DI DATO ASTRATTO IN C

In C, un *ADT* si costruisce definendo:

- ***il nuovo tipo con `typedef`***
- ***una funzione per ogni operazione***

Esempio: il contatore

una entità caratterizzata da un valore intero

```
typedef int counter;
```

con operazioni per

- inizializ. contatore a zero `reset(counter*);`
- incrementare il contatore `inc(counter*);`

ORGANIZZAZIONE DI ADT IN C

La struttura di un ADT comprende quindi:

1. ***un file header,*** contenente
   - `typedef`
   - ***dichiarazione delle funzioni***
2. ***un file di implementazione***, contenente
   - ***direttiva #include per includere il proprio header*** (per importare la definizione di tipo)
   - ***definizione delle funzioni***

ADT `counter`

1. *file header* `counter.h`

```
typedef int counter;
void reset(counter*);
void inc(counter*);
```

Definisce in astratto che cos'è un counter e che cosa si può fare con esso

2. *file di implementazione* `counter.c`

```
#include "counter.h"
void reset(counter *c){ *c=0; }
void inc(counter* c){ (*c)++; }
```

Specifica *come funziona (quale è l'implementazione) di counter*

ADT `counter`: un cliente

Per usare un `counter` occorre:

- ***includere il relativo file header***
- ***definire una o più variabili di tipo `counter`***
- ***operare su tali "oggetti" mediante le sole operazioni (funzioni) previste***

```
#include "counter.h"
int main(){
  counter c1, c2;
  reset(&c1); reset(&c2);
  inc(&c1); inc(&c2); inc(&c2);
}
```

OPERAZIONI DI UN ADT

Quali operazioni definire per un ADT?

- **costruttori** (***costruiscono un oggetto*** di questo tipo, a partire dai suoi "costituenti elementari")
- **selettori** (restituiscono uno dei ***"mattoni elementari"*** che compongono l'oggetto)
- **predicati** (verificano la ***presenza di una proprietà*** sull'oggetto, restituendo *vero* o *falso)*
- **funzioni** (***agiscono*** in vario modo sugli oggetti)
- **trasformatori** (***cambiano lo stato*** dell'oggetto)

ESERCIZIO

Realizzare l'ADT che cattura il concetto di "stringa di al più 250 caratteri"

- **realizzazione basata su un array di caratteri**

Occorre definire le operazioni per:

- estrarre il carattere situato alla i-esima posizione - *SELETTORE*
- calcolare la lunghezza - *FUNZIONE*
- creare una nuova stringa concatenazione di due stringhe date - *COSTRUTTORE*
- confrontare due stringhe - *FUNZIONE*

ESERCIZIO

Realizzare l'ADT che cattura il concetto di "insieme" (di interi)

- **implementazione basata, ad esempio, su una struttura con un array e un indice**

Operazioni per:

- aggiungere un elemento, togliere un elemento - *TRASFORMATORI*
- verificare la presenza di un elemento - *PREDICATO*
- calcolare l'unione di due insiemi, la differenza fra due insiemi, l'intersezione, ecc. - *COSTRUTTORE?*

NOTA: su un insieme non è usualmente definita relazione di ordine e un insieme non può contenere elementi duplicati

ADT IN C: LIMITI

- **Gli ADT così realizzati funzionano, ma *molto dipende dall'autodisciplina del programmatore***
- **Non esiste alcuna protezione contro un uso scorretto dell'ADT**

  l'organizzazione ***suggerisce*** di operare sull'oggetto ***solo tramite le funzioni previste***, ma ***NON riesce a impedire*** di aggirarle a chi lo volesse
  (ad esempio: `counter c1; c1++;`)

- **La struttura interna dell'oggetto *è visibile a tutti*** (nella `typedef`)

ADT IN C: LIMITI

Superare questi limiti sarà uno degli obiettivi cruciali della *programmazione a oggetti*, che vedrete nel corso di Fondamenti di Informatica L-B con il linguaggio Java